NATURAL LANGUAGE INTERACTION WITH SEMANTIC WEB ONTOLOGIES

Gerasimos Lampouras

PH.D. THESIS

DEPARTMENT OF INFORMATICS
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

2015

Abstract

The Semantic Web is an effort to establish standards and mechanisms that will allow computers to reason more easily about the semantics of the Web resources (documents, data etc.). Ontologies play a central role in this endeavor. An ontology provides a conceptualization of a knowledge domain (e.g., consumer electronics) by defining the classes and subclasses of the domain entities, the types of possible relations between them etc. The current standard to specify Semantic Web ontologies is OWL, a formal language based on description logics and RDF, with OWL 2 being the latest OWL standard. Given an OWL ontology for a knowledge domain, one can publish on the Web machine-readable data pertaining to that domain (e.g., catalogues of products, their features etc.), with the data having formally defined semantics based on the conceptualization of the ontology. Several OWL syntaxes have been developed, but people unfamiliar with formal knowledge representation often have difficulties understanding them. This thesis considered methods that allow end-users to view ontology-based knowledge representations of the Semantic Web in the form of automatically generated texts in multiple natural languages.

The first part of the thesis improved NaturalOWL, a Natural Language Generation system for OWL ontologies previously developed at AUEB. The system was modified to support OWL 2 and to be able to produce higher quality texts. Experiments showed that the texts generated by the new version of NaturalOWL are indeed of high quality and significantly better than texts generated by simpler systems, often called ontology

verbalizers, provided that appropriate domain-dependent linguistic resources (e.g., sentence plans to express relations) are available to NaturalOWL.

The second part of the thesis considered text mining and machine learning methods to automatically or semi-automatically extract from the Web the most important of the domain-dependent linguistic resources that NaturalOWL needs to produce high quality texts. Experiments showed that a semi-automatic approach, where a human inspects automatically produced linguistic resources, allows NaturalOWL to produce texts of almost the same quality as with linguistic resources authored manually from scratch.

The third part of the thesis aimed to further improve the quality of the generated texts by developing an Integer Linear Programming model that jointly considers content selection, lexicalization, sentence aggregation, and a limited form of referring expression generation, unlike the pipeline architecture of most natural language generation systems, where the three stages are greedily considered one after the other. Experiments indicated that the new model allows NaturalOWL to express more information per word, which is useful when space is limited (e.g., in advertising), with no deterioration in the perceived quality of the generated texts.

Throughout the thesis, ontologies from different domains (e.g., cultural heritage, consumer electronics, bioinformatics) were used. Using the methods of the thesis, organizations (e.g., companies, libraries, museums) could publish information on the Web both in a machine-readable form (e.g., data originating from databases) and in multiple natural languages (texts automatically generated from data). This way information becomes more easily accessible to both computers and end-users.

Acknowledgements

I would like to thank my supervisor and mentor, Ion Androutsopoulos, for all his guidance, support, and unending patience, throughout the many years of working with him. He always pushed me to strive for excellence, and showed the better way to pursue it. I would also like to thank all the members of the Natural Language Processing Group of AUEB's Department of Informatics for listening to every idea (good or bad) and providing their feedback. I would especially like to thank Makis Malakasiotis, Dimitris Galanis, John Pavlopoulos, and Aris Kosmopoulos for their advice, encouragement, occasional quips and lasting friendship.

My gratitude to all the fantastic people who participated in the human trials of this thesis, too numerous to list here; you know who you are and I thank you. Specifically, I wish to thank Ioanna Giannopoulou for her help in the development of the NaturalOWL Protégé plug-in, and Magda Evaggelakaki for her help in authoring the resources of the Disease Ontology. Also, a big thanks to Yannis Almirantis and Dimitris Polychronopoulos for their annotations and feedback in the experiments of Section 2.4.3.

I thank my parents for always reminding me to face every challenge and potentially regret the outcome, rather than regret not trying, and who supported every decision made in this manner, no matter how bizarre it seemed to them at times. I would be remiss if I did not thank my friends, Fili, Dimitris, Dimitris, and Spyros, for tolerating me and supporting me in any way they could, even if some times no way was evident; their kindness was enough.

Finally, special thanks go to Sofina Pontika for being herself, and providing endless care, understanding and encouragement all these years, from the exciting highs to the brooding lows of research.

This research was co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the "National Strategic Reference Framework (NSRF) – Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund". Especially the experiments with the Disease Ontology (see Sections 2.4.3, 3.4, 4.6.3) were partly funded by the European FP7 ICT project BioASQ.[1]

[1]Consult `http://www.bioasq.org/`.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Generating Texts from OWL Ontologies

The Semantic Web (Berners-Lee et al., 2001; Shadbolt et al., 2006) and the growing popularity of Linked Data (data that are published using Semantic Web technologies) have renewed interest in concept-to-text generation (Reiter and Dale, 2000), especially text generation from ontologies (Bontcheva, 2005; Mellish and Sun, 2006; Galanis and Androutsopoulos, 2007; Mellish and Pan, 2008; Schwitter et al., 2008; Schwitter, 2010a; Liang et al., 2011b; Williams et al., 2011). Ontologies play a central role in the Semantic Web. Each ontology provides a conceptualization of a knowledge domain (e.g., wines, consumer electronics) by defining the classes and subclasses of the individuals (entities) in the domain, the possible relations between them etc.

The current standard to specify ontologies for the Semantic Web is OWL, a formal language based on description logics (Baader et al., 2002), RDF, and RDF SCHEMA, with OWL2 being OWL's latest version (Grau et al., 2008). Given an OWL ontology for a knowledge domain, it is possible to publish machine-readable datasets pertaining to that domain (e.g., catalogues of products, their prices, features etc.) on the Web, with the data having formally defined semantics based on the ontology's conceptualization. It is

also possible to mark up Web resources (e.g., documents) with rich machine-readable meta-data, again with formally defined semantics, to describe their authors, content etc. In both cases, the datasets or meta-data provide instances of the ontology's concepts (e.g., particular individuals of its classes, particular instances of its relation types). Extensions of existing OWL ontologies may also be published, for example to define finer classes, or to combine concepts from several ontologies. Reasoning engines for OWL are also available (Haarslev and Moller, 2001; Tsarkov and Horrocks, 2006; Sirin et al., 2007; Motik et al., 2007), and they can be used, for example, to deduce additional information or to perform consistency checks.

Several equivalent OWL syntaxes have been developed, but people unfamiliar with formal knowledge representation often have difficulties understanding them (Rector et al., 2004). For example, the following statement defines the class of St. Emilion wines, using the functional-style syntax of OWL, one of the easiest to understand.

```
EquivalentClasses(:StEmilion
  ObjectIntersectionOf(:Bordeaux
    ObjectHasValue(:locatedIn :stEmilionRegion)
    ObjectHasValue(:hasColor :red)
    ObjectHasValue(:hasFlavor :strong)
```

The statement above defines `StEmilion` as the intersection of: (i) the class of `Bordeaux` wines; (ii) the class of all individuals whose `locatedIn` property has (for each individual) `stEmilionRegion` among its values (OWL properties are generally many-valued); and (iii), (iv) the classes of individuals whose `hasColor`, and `hasFlavor` property values include `red` and `strong`, respectively, without excluding wines that have additional values in these properties.

Although several *ontology verbalizers* have been developed (Cregan et al., 2007; Kaljurand and Fuchs, 2007; Schwitter et al., 2008; Halaschek-Wiener et al., 2008; Schutte, 2009; Power and Third, 2010; Power, 2010; Schwitter, 2010a; Stevens et al.,

2011; Liang et al., 2011b; Williams et al., 2011) to make ontology statements easier to understand, they usually translate the OWL statements of the ontology one by one to controlled, often not entirely fluent English statements, typically without considering the coherence of the resulting texts, and mostly for the benefit of domain experts. By contrast, this thesis is concerned with producing fluent and coherent multi-sentence texts describing classes or individuals of OWL ontologies, with the texts intended to be read by end-users (e.g., customers of on-line retail sites). For example, we aim to produce texts like the following one, which describes the class of St. Emilion wines defined above:

> St. Emilion is a kind of Bordeaux from the St. Emilion region. It has red color and strong flavor.

The research of this thesis centers around an open-source Natural Language Generation (NLG) system for OWL ontologies, named NaturalOWL, which was previously developed at AUEB (Galanis and Androutsopoulos, 2007; Galanis et al., 2009), but which was significantly improved and extended during the work of this thesis.

1.2 Overview of the thesis

The thesis is organized in three parts. In the first part, we present NaturalOWL version 2, the new version of NaturalOWL that was developed for this thesis. Specifically, we discuss the processing stages of version 2, the optional domain-dependent linguistic resources (e.g., sentence plans to express relations) and user modelling resources that the system employs to produce fluent and coherent texts, the particular NLG issues that arise when generating from OWL ontologies, how version 2 of NaturalOWL compares to the original version, and experiments that assess the quality of the generated texts.

NaturalOWL requires domain-specific linguistic resources to produce high quality texts, but manually constructing these resources can be tedious and costly. In the

second part of this thesis, we consider text mining and machine learning methods to automatically or semi-automatically extract from the Web sentence plans and natural language names, the most important types of domain-specific linguistic resources that NaturalOWL requires. We also present experiments that assess the quality of the automatically or semi-automatically produced linguistic resources and their impact on the generated texts.

In the third and final part, we aim to further improve the quality of the texts that NaturalOWL produces by developing Integer Linear Programming (ILP) models that jointly consider the decisions of content selection, lexicalization, sentence aggregation, and a limited form of referring expression generation. This is contrary to the pipeline architecture of NaturalOWL and most NLG systems(Reiter and Dale, 2000), which considers the decisions of each stage locally and greedily one after the other. We show that our ILP models can produce more compact texts, meaning texts that express more information per word, compared to texts generated using the original pipeline architecture, with no decrease in the perceived quality of the texts. Compact texts of this kind are desirable when space is limited or expensive, for example when displaying product descriptions on smartphones, or when including advertisements in Web search results (Thomaidou et al., 2013; Thomaidou, 2014).

1.3 Contribution of this thesis

The research contribution of each part of the thesis can be summarized as follows.

1.3.1 Natural Language Generation from OWL Ontologies

The first part of the thesis constitutes the first detailed discussion of a complete, general-purpose NLG system for OWL ontologies and the particular issues that arise when generating texts from OWL ontologies. We show through trials with ontologies concerning

different domains (e.g., cultural heritage, consumer electronics, bioinformatics) that a system that relies on NLG methods to a larger extent, compared to simpler OWL verbalizers, can produce significantly better natural language descriptions of classes and individuals, provided that appropriate domain-dependent generation resources are available. We also show how the descriptions can be generated in more than one languages, again provided that appropriate resources are available. Subsequent trials measure the extent to which each processing stage of NLG and each type of domain-dependent generation resources affect the quality of the generated texts, concluding that natural language names, sentence plans, and (to a lesser extent) text plans have the greatest impact on text quality. The new version of NaturalOWL is the first complete, publicly available NLG system for OWL ontologies, excluding much simpler ontology verbalizers. All the domain-dependent linguistic resources of NaturalOWL are also represented in OWL, facilitating their publication and sharing on the Web.

1.3.2 Extracting linguistic resources from the Web

Manually authoring domain-specific generation resources is costly and tedious in most NLG systems. As we demonstrate experimentally, the methods that we propose in the second part of the thesis can be used to construct natural language names and sentence plans, the most important types of domain-specific generation resources, with minimal human involvement, a few hours at most per ontology. By contrast, manually authoring the same resources is typically a matter of several days. Also, no familiarity with OWL and the inner workings of NaturalOWL are needed for an end-user to produce natural language names and sentence plans using the methods of this part of the thesis. The resulting texts are of almost the same quality as when employing manually authored generation resources, and much better than texts produced by using generation resources extracted from the relation and entity identifiers of the ontology (the approach used by simpler verbalizers). Furthermore, unlike previous related work (discussed in

Chapter 3.5), our methods do not require a parallel training corpus of texts and semantic representations; this is particularly important, given that corpora of this kind are very difficult to obtain in practice.

The methods of this part of the thesis have also been embedded in the new version of NaturalOWL. The processing stages and generation resources of NaturalOWL are typical of NLG systems (Reiter and Dale, 2000; Mellish et al., 2006a). Hence, the work of this part should also be applicable, at least in principle, to other NLG systems. Our methods may also be useful in simpler ontology verbalizers, where the main concern seems to be to avoid manually authoring domain-specific linguistic resources, currently at the expense of producing texts of much lower quality.

1.3.3 Generating texts with Integer Linear Programming

The work of the third part of the thesis is the first to consider content selection, lexicalization, and sentence aggregation (and a limited form of referring expression generation) as an ILP joint optimization problem in multi-sentence concept-to-text generation. Previous work in NLG either employed a pipeline architecture with greedy local decisions per processing stage, or considered fewer and different processing stages, was concerned with generating single sentences, or had very different inputs (related work is discussed in Section 4.2).

Experiments with ontologies from different domains (cultural heritage, consumer electronics, bioinformatics) show that our ILP methods manage to produce more compact texts, i.e., to report more facts per word, with no deterioration in the perceived quality of the texts or with improved perceived quality, compared to texts generated by a pipeline architecture. As already noted, compact texts of this kind are desirable when space is limited or expensive, for example when displaying product descriptions on smartphones, or when including advertisements in Web search results. Our experiments also show that our ILP methods, or an approximation that can be used when

longer texts need to be generated, are efficient enough to be used in practice. The ILP methods of this thesis have also been embedded in the new version of NaturalOWL.

1.4 Outline of the remainder of this thesis

The remainder of this thesis is organized as follows: Chapter 2 describes NaturalOWL version 2, its processing stages, its domain-dependent linguistic and user modeling resources, experiments that assess the quality of the generated texts, and the differences between version 2 and the original version of NaturalOWL. Chapter 3 presents our methods to extract domain-specific linguistic resources from the Web, along with experiments that assess the quality of the extracted resources and their impact on the generated texts. Chapter 4 presents our ILP methods, experiments showing that the ILP methods lead to more compact texts, and experiments to assess the efficiency of the ILP methods. Chapter 5 covers more technical details about NaturalOWL and its Protégé plug-in. Chapter 6 concludes and discusses future work.

Chapter 2

Natural Language Generation from OWL Ontologies[1]

2.1 Introduction

Ontologies play a central role in the Semantic Web (Berners-Lee et al., 2001; Shadbolt et al., 2006). Each ontology provides a conceptualization of a knowledge domain (e.g., consumer electronics) by defining the classes and subclasses of the individuals (entities) in the domain, the types of possible relations between them etc. The current standard to specify Semantic Web ontologies is OWL (Horrocks et al., 2003), a formal language based on description logics (Baader et al., 2002), RDF, and RDF SCHEMA (Antoniou and van Harmelen, 2008), with OWL2 being the latest version of OWL (Grau et al., 2008). Given an OWL ontology for a knowledge domain, one can publish, for example, on the Web machine-readable data pertaining to that domain (e.g., catalogues of products, their features etc.), with the data having formally defined semantics based on the con-

[1] An extended version of this chapter, excluding the work of Section 2.4.3, has been published as a journal article (Androutsopoulos et al., 2013).

ceptualization of the ontology.[2] Following common practice in Semantic Web research, we actually use the term *ontology* to refer jointly to *terminological knowledge* (TBox) that establishes a conceptualization of a knowledge domain, and *assertional knowledge* (ABox) that describes particular individuals.

Several equivalent OWL syntaxes have been developed, but people unfamiliar with formal knowledge representation often have difficulties understanding them (Rector et al., 2004). For example, the following statement defines the class of St. Emilion wines, using the functional-style syntax of OWL, one of the easiest to understand, which we also adopt throughout the thesis.[3]

```
EquivalentClasses(:StEmilion
  ObjectIntersectionOf(:Bordeaux
    ObjectHasValue(:locatedIn :stEmilionRegion)
    ObjectHasValue(:hasColor :red)
    ObjectHasValue(:hasFlavor :strong)
    ObjectHasValue(:madeFrom :cabernetSauvignonGrape)
    ObjectMaxCardinality(1 :madeFrom)))
```

To make ontologies easier to understand, several *ontology verbalizers* have been developed (Schwitter, 2010a). Verbalizers usually translate the axioms (in our case, OWL statements) of the ontology one by one to controlled, often not entirely fluent English statements, typically without considering the coherence of the resulting texts, and mostly for the benefit of domain experts. By contrast, in this chapter we present a system that aims to produce fluent and coherent multi-sentence texts describing classes or individuals of OWL ontologies, with the texts intended to be read by end-users (e.g., customers of on-line retail sites). For example, our system can generate the following text from the OWL statement above, if the ontology has been annotated with domain-dependent linguistic resources discussed below.

[2] See `http://owl.cs.manchester.ac.uk/repository/` for a repository of OWL ontologies.

[3] Consult `http://www.w3.org/TR/owl2-primer/` for an introduction to the functional-style syntax of OWL.

St. Emilion is a kind of Bordeaux from the St. Emilion region. It has red color and strong flavor. It is made from exactly one grape variety: Cabernet Sauvignon grapes.

Our system, called NaturalOWL, is open-source and supports both English and Greek. Hence, Greek texts can also be generated from the same OWL statements, as in the following product description, provided that appropriate Greek linguistic resources are also available. By contrast, OWL verbalizers typically produce only English (or English-like) sentences.

```
ClassAssertion(:Laptop :tecraA8)
ObjectPropertyAssertion(:manufacturedBy :tecraA8 :toshiba)
ObjectPropertyAssertion(:hasProcessor :tecraA8 :intelCore2)
DataPropertyAssertion(:hasMemoryInGB :tecraA8 "2"^^xsd:nonNegativeInteger)
DataPropertyAssertion(:hasHardDiskInGB :tecraA8 "110"^^xsd:nonNegativeInteger)
DataPropertyAssertion(:hasSpeedInGHz :tecraA8 "2"^^xsd:float)
DataPropertyAssertion(:hasPriceInEuro :tecraA8 "850"^^xsd:nonNegativeInteger)
```

[English description:] Tecra A8 is a laptop, manufactured by Toshiba. It has an Intel Core 2 processor, 2 GB RAM and a 110 GB hard disk. Its speed is 2 GHz and it costs 850 Euro.

[Greek description:] Ο Tecra A8 είναι ένας φορητός υπολογιστής, κατασκευασμένος από την Toshiba. Διαθέτει επεξεργαστή Intel Core 2, 2 GB RAM και σκληρό δίσκο 110 GB. Η ταχύτητά του είναι 2 GHz και κοστίζει 850 Ευρώ.

The examples above illustrate how a system like NaturalOWL can help publish information on the Web both as OWL statements and as texts generated from the OWL statements. This way, information becomes easily accessible to both computers, which can process the OWL statements, and end-users speaking different languages; and changes in the OWL statements can be automatically reflected in the texts by regenerating them. To produce fluent, coherent multi-sentence texts, NaturalOWL relies on natural language generation (NLG) methods (McKeown, 1985; Reiter and Dale, 2000) to a larger extent compared to existing OWL verbalizers; for example, it includes mechanisms to avoid

repeating information, to order the facts to be expressed, aggregate smaller sentences into longer ones, generate referring expressions etc. Although NLG is an established area, we are the first to discuss in detail an NLG system for OWL ontologies, excluding simpler verbalizers. We do not propose novel algorithms from a theoretical NLG perspective in this chapter, but we show that there are several particular issues that need to be considered when generating texts from OWL ontologies. For example, some OWL statements lead to overly complicated sentences, unless they are converted to simpler intermediate representations first; there are also several OWL-specific opportunities to aggregate sentences (e.g., when expressing axioms about the cardinalities of properties); and referring expression generation can exploit the class hierarchy.

NaturalOWL can be used with any OWL ontology, but to obtain texts of high quality *domain-dependent generation resources* are required; for example, the classes of the ontology can be mapped to natural language names, the properties to sentence plans etc. Similar linguistic resources are used in most NLG systems, though different systems adopt different linguistic theories and algorithms, requiring different resources. There is little consensus on exactly what information NLG resources should capture, apart from abstract specifications (Mellish, 2010). The domain-dependent generation resources of NaturalOWL are created by a *domain author*, a person familiar with OWL, when the system is configured for a new ontology. The domain author uses the Protégé ontology editor and a Protégé plug-in that allows editing the domain-dependent generation resources and invoking NaturalOWL to view the resulting texts.[4]

OWL ontologies often use English words or concatenations of words as identifiers of classes, properties, and individuals (e.g., `manufacturedBy`). Hence, some of the domain-dependent generation resources can often be extracted from the ontology by guessing, for example, that a class identifier like `Laptop` in our earlier ex-

[4]Consult `http://protege.stanford.edu/` for information on Protégé. NaturalOWL and its Protégé plug-in are freely available from `http://nlp.cs.aueb.gr/software.html`.

ample is a noun that can be used to refer to that class, or that a statement of the form `ObjectPropertyAssertion(:manufacturedBy X Y)` should be expressed in English as a sentence of the form "*X* was manufactured by *Y*". Most OWL verbalizers follow this strategy. Similarly, if domain-dependent generation resources are not provided, NaturalOWL attempts to extract them from the ontology, or it uses generic resources. The resulting texts, however, are of lower quality; also, non-English texts cannot be generated, if the identifiers of the ontology are English-like. There is a tradeoff between reducing the effort to construct domain-dependent generation resources for OWL ontologies, and obtaining higher-quality texts in multiple languages, but this tradeoff has not been investigated in previous work. We present trials we performed to measure the effort required to construct the domain-dependent generation resources of NaturalOWL and the extent to which they improve the resulting texts, also comparing against a simpler verbalizer that requires no domain-dependent generation resources. The trials show that the domain-dependent generation resources help NaturalOWL produce significantly better texts, and that the resources can be constructed with relatively light effort, compared to the effort typically needed to construct an ontology, though several days may be needed to construct the resources manually. In Chapter 3, we discuss additional methods that can be used to automatically or semi-automatically extract the domain-dependent generation resources of NaturalOWL from the Web.

Overall, the main contributions of this chapter are: (i) it is the first detailed discussion of a complete, general-purpose NLG system for OWL ontologies and the particular issues that arise when generating texts from OWL ontologies; (ii) it shows that a system that relies on NLG methods to a larger extent, compared to simpler OWL verbalizers, can produce significantly better natural language descriptions of classes and individuals, provided that appropriate domain-dependent generation resources are available; (iii) it shows how the descriptions can be generated in more than one languages, again provided that appropriate resources are available; (iv) it shows that the domain-dependent

generation resources can be constructed with relatively light, but not negligible effort, compared to the effort needed to construct an ontology. As already noted, this chapter does not present novel algorithms from a theoretical NLG perspective. In fact, some of the algorithms that NaturalOWL uses are of a narrower scope, compared to more fully-fledged NLG algorithms. Nevertheless, the trials show that the system produces texts of reasonable quality, especially when domain-dependent generation resources are provided. We hope that if NaturalOWL contributes towards a wider adoption of NLG methods on the Semantic Web, other researchers may wish to contribute improved components, given that NaturalOWL is open-source.

NaturalOWL is based on ideas from ILEX (O'Donnell et al., 2001) and M-PIRO (Isard et al., 2003). The ILEX project developed an NLG system that was demonstrated mostly with museum exhibits, but did not support OWL.[5] The M-PIRO project produced a multilingual extension of the system of ILEX, which was tested in several domains (Androutsopoulos et al., 2007). Attempts to use the generator of M-PIRO with OWL, however, ran into problems (Androutsopoulos et al., 2005). By contrast, NaturalOWL was especially developed for OWL. We describe version 2 of NaturalOWL in this chapter, which is the version developed during the work of this thesis. Version 1 of (Galanis and Androutsopoulos, 2007) was much more limited. The main differences between the two versions are summarized at the end of this chapter (Section 2.5). The NaturalOWL plug-in of NaturalOWL version 2, which was also developed during the work of this thesis, and its usage are discussed in more detail in Chapter 5.

In the remainder of this chapter and thesis overall, we assume that the reader is familiar with RDF, RDF SCHEMA, and OWL. Readers unfamiliar with the Semantic Web may wish to consult an introductory text first (Antoniou and van Harmelen, 2008). We also note that the recently very popular Linked Data are published and interconnected

[5] Dale et al. (1998) and Dannels (2008; 2012) also discuss NLG for museums.

using Semantic Web technologies.[6] Most Linked Data currently use only RDF and RDF SCHEMA, but OWL is in effect a superset of RDF SCHEMA and, hence, the work of this chapter also applies to Linked Data.

Section 2.2 below briefly discusses some related work; we provide further pointers to related work in subsequent sections. Section 2.3 then explains how NaturalOWL generates texts, also discussing the domain-dependent generation resources of each processing stage. Section 2.4 describes the trials we performed to measure the effort required to construct the domain-dependent generation resources and their impact on the quality of the generated texts. Section 2.5 highlights the main differences between NaturalOWL version 1 and version 2. Section 2.6 concludes.

2.2 Related work

We use the functional-style syntax of OWL in this chapter, but several equivalent OWL syntaxes exist.[7] There has also been work to develop controlled natural languages (CNLs), mostly English-like, to be used as alternative OWL syntaxes. Sydney OWL Syntax (SOS) (Cregan et al., 2007) is an English-like CNL with a bidirectional mapping to and from the functional-style syntax of OWL; SOS is based on PENG (Schwitter and Tilbrook, 2004). A similar bidirectional mapping has been defined for Attempto Controlled English (ACE) (Kaljurand, 2007). Rabbit (Denaux et al., 2010) and CLOnE (Funk et al., 2007) are other OWL CNLs, mostly intended to be used by domain experts when authoring ontologies (Denaux et al., 2011). We also note that some OWL CNLs cannot express all the kinds of OWL statements (Schwitter et al., 2008).

[6] Consult `http://linkeddata.org/`. See also the work of Duma and Klein (2013).

[7] Other available OWL syntaxes include the following: the RDF syntax (`http://www.w3.org/TR/2009/REC-owl2-mapping-to-rdf-20091027/`), the OWL/XML syntax (`http://www.w3.org/TR/2009/REC-owl2-xml-serialization-20091027/`), and the Manchester OWL syntax (`http://www.w3.org/TR/2009/NOTE-owl2-manchester-syntax-20091027/`).

Much work on OWL CNLs focuses on ontology authoring and querying (Bernardi et al., 2007; Kaufmann and Bernstein, 2010; Schwitter, 2010b); the emphasis is mostly on the direction from CNL to OWL or query languages.[8] More relevant to our work are CNLs like SOS and ACE, to which automatic mappings from normative OWL syntaxes are available. By feeding an OWL ontology expressed, for example, in functional-style syntax to a mapping that translates to an English-like CNL, all the axioms of the ontology can be turned into English-like sentences. Systems of this kind are often called *ontology verbalizers*. This term, however, also includes systems that translate from OWL to English-like statements that do not belong in an explicitly defined CNL (Halaschek-Wiener et al., 2008; Schutte, 2009; Power and Third, 2010; Power, 2010; Stevens et al., 2011; Liang et al., 2011b).

Although verbalizers can be viewed as performing a kind of light NLG, they typically translate axioms one by one, as already noted, without considering the coherence (or topical cohesion) of the resulting texts, usually without aggregating sentences nor generating referring expressions, and often by producing sentences that are not entirely fluent or natural. For example, ACE and SOS occasionally use variables instead of referring expressions (Schwitter et al., 2008). Also, verbalizers typically do not employ domain-dependent generation resources and typically do not support multiple languages. Expressing the exact meaning of the axioms of the ontology in an unambiguous manner is considered more important in verbalizers than composing a fluent and coherent text in multiple languages, partly because the verbalizers are typically intended to be used by domain experts.

Some verbalizers use ideas and methods from NLG. For example, some verbalizers include sentence aggregation (Williams and Power, 2010) and text planning (Liang

[8] *Conceptual authoring* or WYSIWYM (Power and Scott, 1998; Hallett et al., 2007), which has been applied to OWL (Power, 2009), and *round-trip authoring* (Davis et al., 2008) are bidirectional, but focus mostly on ontology authoring and querying.

et al., 2011a). Overall, however, NLG methods have been used only to a very limited extent with OWL ontologies. A notable exception is ONTOSUM (Bontcheva, 2005), which generates natural language descriptions of individuals, but apparently not classes, from RDF SCHEMA and OWL ontologies. It is an extension of MIAKT (Bontcheva and Wilks, 2004), which was used to generate medical reports. Both were implemented in GATE (Bontcheva et al., 2004) and they provide graphical user interfaces to manipulate domain-dependent generation resources (Bontcheva and Cunningham, 2003). No detailed description of ONTOSUM appears to have been published, however, and the system does not seem to be publicly available, unlike NaturalOWL. Also, no trials of ONTOSUM with independently created ontologies seem to have been published.

Mellish and Sun (2006) focus on lexicalization and sentence aggregation, aiming to produce a single aggregated sentence from an input collection of RDF triples; by contrast, NaturalOWL produces multi-sentence texts. In complementary work, Mellish et al. (2008) consider content selection for texts describing OWL classes. Unlike NaturalOWL, their system does not express only facts that are explicit in the ontology, but also facts deduced from the ontology. Nguyen et al. (2012) discuss how the proof trees of facts deduced from OWL ontologies can be explained in natural language. It would be particularly interesting to examine how deduction and explanation mechanisms could be added to NaturalOWL.

2.3 Processing stages and resources of NaturalOWL

NaturalOWL adopts a pipeline architecture, which is common in NLG (Reiter and Dale, 2000), though the number and purpose of its components often vary (Mellish et al., 2006b). Our system generates texts in three stages, *document planning*, *micro-planning*, and *surface realization*, discussed in the following sections; see Figure 2.1.

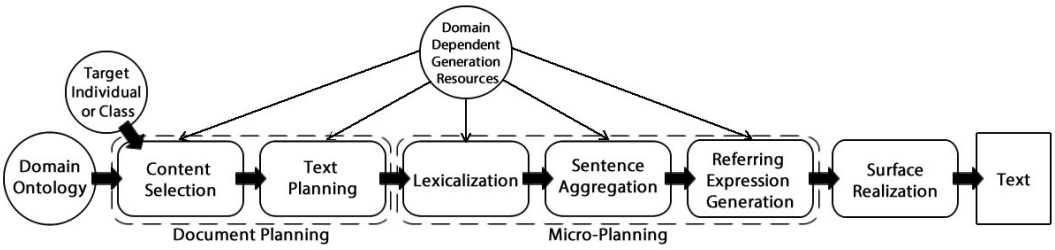


Figure 2.1: The processing stages and sub-stages of NaturalOWL.

2.3.1 Document Planning

Document planning consists of *content selection*, where the system selects the information to convey, and *text planning*, where it plans the structure of the text to be generated.

2.3.1.1 Content Selection

In content selection, the system first retrieves from the ontology all the OWL statements that are relevant to the class or individual to be described, it then converts the selected OWL statements to message triples, which are easier to express as sentences, and it finally selects among the message triples the ones to be expressed.

OWL statements for individual targets

Let us first consider content selection when NaturalOWL is asked to describe an *individual* (an entity), and let us call that individual the *target*. The system scans the OWL statements of the ontology, looking for statements of the forms listed in the left column of Table 2.1.[9] In effect, it retrieves all the statements that describe the target directly, as opposed to statements describing another individual or a (named) class the target is related to.

The OWL language specification allows arbitrarily many nested `ObjectUnionOf` and `ObjectIntersectionOf` operators, which may lead to statements that are very

[9]Some OWL statements shown in Table 2.1 with two arguments can actually have more arguments, but they can be converted to the forms shown.

OWL statements	Message triples
ClassAssertion(*NamedClass target*)	<*target*, instanceOf, *NamedClass*>
ClassAssertion(ObjectComplementOf(*NamedClass*) *target*)	<*target*, not(instanceOf), *NamedClass*>
ClassAssertion(ObjectOneOf(*indiv1 indiv2 ...*) *target*)	<*target*, oneOf, or(*indiv1*, *indiv2*, ...)>
ClassAssertion(ObjectHasValue(*objProp indiv*) *target*)	<*target*, *objProp*, *indiv*>
ClassAssertion(ObjectHasValue(*dataProp dataValue*) *target*)	<*target*, *dataProp*, *dataValue*>
ClassAssertion(ObjectHasSelf(*objProp*) *target*)	<*target*, *objProp*, *target*>
ClassAssertion(ObjectMaxCardinality(*number prop* [*NamedClass*]) *target*)	<*target*, maxCardinality(*prop*), *number*[:*NamedClass*]>
ClassAssertion(ObjectMinCardinality(*number prop* [*NamedClass*]) *target*)	<*target*, minCardinality(*prop*), *number*[:*NamedClass*]>
ClassAssertion(ObjectExactCardinality(*number prop* [*NamedClass*]) *target*)	<*target*, exactCardinality(*prop*), *number*[:*NamedClass*]>
ClassAssertion(ObjectSomeValuesFrom(*objProp NamedClass*) *target*)	<*target*, someValuesFrom(*objProp*), *NamedClass*>
ClassAssertion(ObjectAllValuesFrom(*objProp NamedClass*) *target*)	<*target*, allValuesFrom(*objProp*), *NamedClass*>
ClassAssertion(ObjectIntersectionOf(*C1 C2 ...*) *target*)	*convert*(ClassAssertion(*C1 target*)) *convert*(ClassAssertion(*C2 target*)) ...
ClassAssertion(ObjectUnionOf(*C1 C2 ...*) *target*)	or(*convert*(ClassAssertion(*C1 target*)), *convert*(ClassAssertion(*C2 target*)), ...)
ObjectPropertyAssertion(*objProp target indiv*)	<*target*, *objProp*, *indiv*>
DataPropertyAssertion(*dataProp target dataValue*)	<*target*, *dataProp*, *dataValue*>
NegativeObjectPropertyAssertion(*objProp target indiv*)	<*target*, not(*objProp*), *indiv*>
NegativeDataPropertyAssertion(*dataProp target dataValue*)	<*target*, not(*dataProp*), *dataValue*>
DifferentIndividuals(*target indiv*)	<*target*, differentIndividuals, *indiv*>
DifferentIndividuals(*indiv target*)	<*target*, differentIndividuals, *indiv*>
SameIndividual(*target indiv*)	<*target*, sameIndividual, *indiv*>
SameIndividual(*indiv target*)	<*target*, sameIndividual, *indiv*>

Notation: Square brackets indicate optional arguments, and *convert*(ξ) a recursive application of the conversion to ξ. *NamedClass* is a class identifier; *objProp*, *dataProp*, and *prop* are identifiers of object properties, datatype properties, and properties; *indiv*, *indiv1*, ... are identifiers of individuals; *dataValue* is a datatype value; and *C*, *C1*, ... are class identifiers, or expressions constructing classes without ObjectIntersectionOf or ObjectUnionOf.

Table 2.1: OWL statements for an individual target, and corresponding message triples.

difficult to express in natural language. To simplify text generation and to ensure that the resulting texts are easy to comprehend, we do not allow nested `ObjectUnionOf` and `ObjectIntersectionOf` operators in the ontologies the texts are generated from. In Table 2.1, this restriction is enforced by requiring class identifiers to appear at some points where OWL also allows expressions that construct unnamed classes using operators. If an ontology uses unnamed classes at points where Table 2.1 requires class identifiers (named classes), it can be easily modified to comply with Table 2.1 by defining new named classes for nested unnamed ones.[10] In practice, nested `ObjectUnionOf` and `ObjectIntersectionOf` operators are rare; see the work of Power et al. (Power, 2010; Power and Third, 2010; Power, 2012) for information on the frequencies of different types of OWL statements.[11]

Statements of the form `ClassAssertion(`*`Class target`*`)` may be quite complex, because *`Class`* is not necessarily a class identifier. It may also be an expression constructing an unnamed class, as in the following example. This is why there are multiple rows for `ClassAssertion` in Table 2.1.

```
ClassAssertion(
  ObjectIntersectionOf(:Wine
    ObjectHasValue(:locatedIn :stEmilionRegion)
    ObjectHasValue(:hasColor :red)
    ObjectHasValue(:hasFlavor :strong)
    ObjectHasValue(:madeFrom :cabernetSauvignonGrape)
```

[10] It is also easy to automatically detect nested unnamed classes and replace them, again automatically, by new named classes (classes with OWL identifiers). The domain author would have to be consulted, though, to provide meaningful OWL identifiers for the new classes (otherwise arbitrary identifiers would have to be used) and natural language names for the new classes (see Section 2.3.2.1 below).

[11] One could also refactor some nested operators; for example, $t \in ((A \cup B) \cap (C \cup D))$ is equivalent to $t \in (A \cup B)$ and $t \in (C \cup D)$. The conversion to message triples, to be discussed below, in effect also performs some refactoring, but it cannot cope with all the possible nested union and intersection operators, which is why we disallow them as a general rule.

```
      ObjectMaxCardinality(1 :madeFrom))
    :chateauTeyssier2007)
```

NaturalOWL would express the OWL statement above by generating the following text.

> The 2007 Chateau Teyssier is a wine from the St. Emilion region. It has red color and strong flavor. It is made from exactly one grape variety: Cabernet Sauvignon grapes.

Recall that the texts of NaturalOWL are intended to be read by end-users. Hence, we prefer to generate texts that may not emphasize enough some of the subtleties of the OWL statements, in order to produce more readable texts. An OWL expert might prefer, for example, the following description of `chateauTeyssier2007`, which mirrors more closely the corresponding OWL statements.

> The 2007 Chateau Teyssier is a member of the intersection of: (a) the class of wines, (b) the class of individuals from (not necessarily exclusively) the St. Emilion region, (c) the class of individuals that have (not necessarily exclusively) red color, (d) the class of individuals that have (not necessarily exclusively) strong flavor, (e) the class of individuals that are made exclusively from Cabernet Sauvignon grapes.

Stricter texts of this kind, however, seem inappropriate for end-users. In fact, it could be argued that even mentioning that the wine is made from *exactly* one grape variety in the produced text is inappropriate for end-users. Our system can be instructed to avoid mentioning this information via user modelling annotations, discussed below.

OWL statements for class targets

If the system is asked to describe a *class*, rather than an individual, it scans the ontology for statements of the forms listed in the left column of Table 2.2. The class to be described must be a named one, meaning that it must have an OWL identifier, and `Target` denotes its identifier. Again, to simplify the generation process and to avoid producing complicated texts, Table 2.2 requires class identifiers to appear at some points

where OWL also allows expressions that construct unnamed classes using operators. If an ontology uses unnamed classes at points where Table 2.2 requires class identifiers, it can be easily modified.

In texts describing classes, it is difficult to express informally the difference between `EquivalentClasses` and `SubClassOf`. `EquivalentClasses(`*`C1 C2`*`)` means that any individual of *C1* also belongs in *C2*, and vice versa. By contrast, `SubClassOf(`*`C1 C2`*`)` means that any member of *C1* also belongs in *C2*, but the reverse is not necessarily true. If we replace `EquivalentClasses` by `SubClassOf` in the definition of `StEmilion` of page 9, any member of `StEmilion` is still necessarily also a member of the intersection, but a wine with all the characteristics of the intersection is not necessarily a member of `StEmilion`. Consequently, one should perhaps add sentences like the ones shown in italics below, when expressing `EquivalentClasses` and `SubClassOf`, respectively.

> St. Emilion is a kind of Bordeaux from the St. Emilion region. It has red color and strong flavor. It is made from exactly one grape variety: Cabernet Sauvignon grapes. *Every St. Emilion has these properties, and anything that has these properties is a St. Emilion.*

> St. Emilion is a kind of Bordeaux from the St. Emilion region. It has red color and strong flavor. It is made from exactly one grape variety: Cabernet Sauvignon grapes. *Every St. Emilion has these properties, but something may have these properties without being a St. Emilion.*

NaturalOWL produces the same exact texts, without the sentences in italics, for both `SubClassOf` and `EquivalentClasses`, to avoid generating texts that sound too formal. Also, it may not mention some of the information of the ontology about a target class (e.g., that a St. Emilion has strong flavor), when user modelling indicates that this information is already known or that the text should not exceed a particular length. Hence, the generated texts express *necessary*, not *sufficient* conditions for individuals to belong in the target class.

OWL statements	Message triples
EquivalentClasses(*Target C*)	*convert*(SubClassOf(*Target C*))
EquivalentClasses(*C Target*)	*convert*(SubClassOf(*Target C*))
SubClassOf(*Target NamedClass*)	<*Target*, isA, *NamedClass*>
SubClassOf(*Target* ObjectComplementOf(*NamedClass*))	<*Target*, not(isA), *NamedClass*>
SubClassOf(*Target* ObjectOneOf(*indiv1 indiv2 ...*))	<*Target*, oneOf, or(*indiv1*, *indiv2*, ...)>
SubClassOf(*Target* ObjectHasValue(*objProp indiv*))	<*Target*, *objProp*, *indiv*>
SubClassOf(*Target* ObjectHasValue(*dataProp dataValue*))	<*Target*, *dataProp*, *dataValue*>
SubClassOf(*Target* ObjectHasSelf(*objProp*))	<*Target*, *objProp*, *Target*>
SubClassOf(*Target* ObjectMaxCardinality(*number prop* [*NamedClass*]))	<*Target*, maxCardinality(*prop*), *number*[:*NamedClass*]>
SubClassOf(*Target* ObjectMinCardinality(*number prop* [*NamedClass*]))	<*Target*, minCardinality(*prop*), *number*[:*NamedClass*]>
SubClassOf(*Target* ObjectExactCardinality(*number prop* [*NamedClass*]))	<*Target*, exactCardinality(*objProp*), *number*[:*NamedClass*]>
SubClassOf(*Target* ObjectSomeValuesFrom(*objProp NamedClass*))	<*Target*, someValuesFrom(*objProp*), *NamedClass*>
SubClassOf(*Target* ObjectAllValuesFrom(*objProp NamedClass*))	<*Target*, allValuesFrom(*objProp*), *NamedClass*>
SubClassOf(*Target* ObjectIntersectionOf(*C1 C2 ...*))	*convert*(SubClassOf(*C1 Target*)) *convert*(SubClassOf(*C2 Target*)) ...
SubClassOf(*Target* ObjectUnionOf(*C1 C2 ...*))	or(*convert*(SubClassOf(*C1 Target*)), *convert*(SubClassOf(*C2 Target*)), ...)
DisjointClasses(*Target NamedClass*)	<*Target*, not(isA), *NamedClass*>
DisjointClasses(*NamedClass Target*)	<*Target*, not(isA), *NamedClass*>

Notation: Square brackets indicate optional arguments, and *convert*(ξ) a recursive application of the conversion to ξ. *NamedClass* is a class identifier; *objProp*, *dataProp*, and *prop* are identifiers of object properties, datatype properties, and properties; *indiv*, *indiv1*, ... are identifiers of individuals; *dataValue* is a datatype value; and *C*, *C1*, ... are class identifiers, or expressions constructing classes without ObjectIntersectionOf or ObjectUnionOf.

Table 2.2: OWL statements for a class target, and the corresponding message triples.

OWL statements for second-level targets

In some applications, expressing additional OWL statements that are *indirectly* related to the target may be desirable. Let us assume, for example, that the target is the individual `exhibit24`, and that the following directly relevant statements have been retrieved from the ontology. NaturalOWL would express them by generating a text like the one below.

```
ClassAssertion(:Aryballos :exhibit24)
ObjectPropertyAssertion(:locationFound :exhibit24 :heraionOfDelos)
ObjectPropertyAssertion(:creationPeriod :exhibit24 :archaicPeriod)
ObjectPropertyAssertion(:paintingTechniqueUsed :exhibit24 :blackFigureTechnique)
ObjectPropertyAssertion(:currentMuseum :exhibit24 :delosMuseum)
```

This is an aryballos, found at the Heraion of Delos. It was created during the archaic period and it was decorated with the black-figure technique. It is currently in the Museum of Delos.

The names of classes and individuals can be shown as hyperlinks to indicate that they can be used as subsequent targets. Clicking on a hyperlink would be a request to generate a description for the corresponding class or individual. Alternatively, we may retrieve in advance the OWL statements for the subsequent targets and add them to those of the current target.

More precisely, assuming that the target is an individual, the subsequent targets, called *second-level targets*, are the target's class, provided that it is a named one, and the individuals the target is directly linked to via object properties. NaturalOWL considers second-level targets only when the current target is an individual, because with class targets, second-level targets often lead to complicated texts. To retrieve OWL statements for both the current and the second-level targets (when applicable), or only for the current target, we set the *maximum fact distance* to 2 or 1, respectively. Returning to `exhibit24`, let us assume that the maximum fact distance is 2 and that the following OWL statements for second-level targets have been retrieved.[12]

[12]Consult `http://www.w3.org/TR/owl-time/` for more principled representations of time in OWL.

```
SubClassOf(:Aryballos :Vase)
SubClassOf(:Aryballos
  ObjectHasValue(:exhibitTypeCannedDescription
    "An aryballos was a small spherical vase with a narrow neck, in which the
    athletes kept the oil they spread their bodies with"^^xsd:string))
DatatypePropertyAssertion(:periodDuration :archaicPeriod
    "700 BC to 480 BC"^^xsd:string)
DatatypePropertyAssertion(:periodCannedDescription :archaicPeriod
    "The archaic period was when the Greek ancient city-states
    developed"^^xsd:string)
DataPropertyAssertion(:techniqueCannedDescription :blackFigureTechnique
    "In the black-figure technique, the silhouettes are rendered in black on the
    pale surface of the clay, and details are engraved"^^xsd:string)
```

To express all the retrieved OWL statements, including those for the second-level targets, NaturalOWL would now generate a text like the following, which may be preferable, if this is the first time the user encounters an aryballos and archaic exhibits.

> This is an aryballos, a kind of vase. An aryballos was a small spherical vase with a narrow neck, in which the athletes kept the oil they spread their bodies with. This aryballos was found at the Heraion of Delos and it was created during the archaic period. The archaic period was when the Greek ancient city-states developed and it spans from 700 BC to 480 BC. This aryballos was decorated with the black-figure technique. In the black-figure technique, the silhouettes are rendered in black on the pale surface of the clay, and details are engraved. This aryballos is currently in the Museum of Delos.

We note that in many ontologies it is impractical to represent all the information in logical terms. In our example, it is much easier to store the information that "An aryballos was a small. . . bodies with" as a string, i.e., as a *canned* sentence, rather than defining classes, properties, and individuals for spreading actions, bodies, etc. and generating the sentence from a logical meaning representation. Canned sentences, however, have to be entered in multiple versions, if several languages or user types need to be supported.

Converting OWL statements to message triples

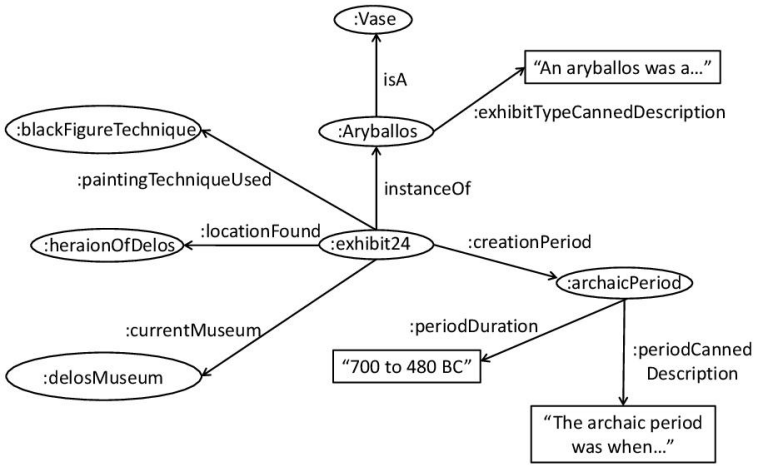


Figure 2.2: Graph view of message triples.

Tables 2.1 and 2.2 also show how the retrieved OWL statements can be rewritten as triples of the form $\langle S, P, O \rangle$, where S is the target or a second-level target; O is an individual, datatype value, class, or a set of individuals, datatype values, or classes that S is mapped to; and P specifies the kind of mapping. We call S the *semantic subject* or *owner* of the triple, and O the *semantic object* or *filler*; the triple can also be viewed as a field named P, owned by S, and filled by O. For example, the OWL statements about `exhibit24` shown above, including those about the second-level targets, are converted to the following triples.

```
<:exhibit24, instanceOf, :Aryballos>
<:exhibit24, :locationFound, :heraionOfDelos>
<:exhibit24, :creationPeriod, :archaicPeriod>
<:exhibit24, :paintingTechniqueUsed, :blackFigureTechnique>
<:exhibit24, :currentMuseum, :delosMuseum>
<:Aryballos, isA, :Vase>
<:Aryballos, :exhibitTypeCannedDescription,
    "An aryballos was a... bodies with"^^xsd:string>
<:archaicPeriod, :periodDuration, "700 BC to 480 BC"^^xsd:string>
<:archaicPeriod, :periodCannedDescription,
    "The archaic period was..."^^xsd:string>
<:blackFigureTechnique, :techniqueCannedDescription,
    "In the black-figure..."^^xsd:string>
```

More precisely, P can be: (i) a property of the ontology; (ii) one of the keywords `isA`, `instanceOf`, `oneOf`, `differentIndividuals`, `sameIndividuals`; or (iii) an expression of the form *modifier*(ρ), where *modifier* may be `not`, `maxCardinality` etc. (see Tables 2.1 and 2.2) and ρ is a property of the ontology. We hereafter call *properties* all three types of P, though types (ii) and (iii) are strictly not properties in the terminology of OWL. If we need to distinguish between them, we use the terms *property of the ontology*, *domain-independent property*, and *modified property*, respectively.

Every OWL statement or collection of OWL statements can be represented as a set of RDF triples.[13] The triples of Tables 2.1–2.2 are similar, but not the same as RDF triples. Most notably, expressions of the form *modifier*(ρ) cannot be used as P in RDF triples. To avoid confusion, we call *message triples* the triples of Tables 2.1–2.2, to distinguish them from RDF triples. As with RDF triples, message triples can be viewed as forming a graph. Figure 2.2 shows the graph for the message triples of `exhibit24`; the triple linking `blackFigureTechnique` to a canned sentence is not shown to save space. The second-level targets are the classes and individuals at distance one from the target (`exhibit24`).[14] By contrast, the graph for the RDF triples representing the OWL statements would be more complicated, and second-level targets would not always be at distance one from the target.

Each message triple is intended to be easily expressible as a simple sentence, which is not always the case with RDF triples representing OWL statements. The message triples also capture similarities of the sentences to be generated that may be less obvious when looking at the original OWL statements or the RDF triples representing them. For example, the `ClassAssertion` and `SubClassOf` statements below are mapped to identical message triples, apart from the identifiers of the individual and the class, and

[13] See `http://www.w3.org/TR/owl2-mapping-to-rdf/`.

[14] Instead of retrieving the OWL statements about the target and second-level targets and then converting them to message triples, one could equivalently convert all the OWL statements of the ontology to message triples and select the message triples connecting the target to nodes up to distance two from the target.

the similarity of the message triples reflects the similarity of the resulting sentences, also shown below.

```
ClassAssertion(ObjectMaxCardinality(1 :madeFromGrape) :product145)

<:product145, maxCardinality(:madeFromGrape), 1>
```

Product 145 is made from at most one grape.

```
SubClassOf(:StEmilion ObjectMaxCardinality(1 :madeFromGrape))

<:StEmilion, maxCardinality(:madeFromGrape), 1>
```

St. Emilion is made from at most one grape.

By contrast, without the conversion to message triples, the OWL statements and the RDF triples representing them would lead to more difficult to follow sentences like the following:

Product 145 is a member of the class of individuals that are made from at most one grape.

St. Emilion is a subclass of the class of individuals that are made from at most one grape.

As a further example, Tables 2.1 and 2.2 discard `ObjectIntersectionOf` , producing multiple message triples instead. For example, the `EquivalentClasses` statement defining `StEmilion` on page 9 would be converted to the following message triples.

```
<:StEmilion, isA, :Bordeaux>
<:StEmilion, :locatedIn, :stEmilionRegion>
<:StEmilion, :hasColor, :red>
<:StEmilion, :hasFlavor, :strong>
<:StEmilion, :madeFromGrape, :cabernetSauvignonGrape>
<:StEmilion, maxCardinality(:madeFromGrape), 1>
```

The resulting message triples correspond to the sentences below, where subsequent references to `StEmilion` have been replaced by pronouns to improve readability; the sentences could also be aggregated into longer ones, as discussed in later sections.

> St. Emilion is a kind of Bordeaux. It is from the St. Emilion region. It has red color. It has strong flavor. It is made from Cabernet Sauvignon grape. It is made from at most one grape variety.

By contrast the original OWL statement of page 9 and the RDF triples representing it would lead to the 'stricter' text of page 20, which is inappropriate for end-users, as already noted. Notice, also, that Table 2.2 converts `EquivalentClasses` and `SubClassOf` statements to identical triples, where *P* is `isA`, since NaturalOWL produces the same texts for both kinds of statements, as already discussed.

Tables 2.1 and 2.2 also replace `ObjectUnionOf` operators by disjunctions of message triples. The following OWL statement is mapped to the message triple shown below:

```
ClassAssertion(
 UnionOf(ObjectHasValue(:hasFlavor :strong) ObjectHasValue(:hasFlavor :medium))
 :houseWine)

or(<:houseWine, :hasFlavor, :strong>, <:houseWine, :hasFlavor, :medium>)
```

which leads to the first sentence below; the sentence can then be shortened during aggregation, leading to the second sentence below.

> The house wine has strong flavor or it has medium flavor.
>
> The house wine has strong or medium flavor.

By contrast, the OWL statement and the corresponding RDF triples in effect say that:

> The house wine is a member of the union of: (i) the class of all wines that have strong flavor, and (ii) the class of all wines that have medium flavor.

Interest scores and repetitions

Expressing all the message triples of all the retrieved OWL statements is not always appropriate. Let us assume, for example, that the maximum fact distance is 2 and that a description of `exhibit24` of Figure 2.2 has been requested by a museum visitor. It may be the case that the visitor has already encountered other archaic exhibits, and that the duration of the archaic period was mentioned in previous descriptions. Repeating the duration of the period may, thus, be undesirable. We may also want to exclude message triples that are uninteresting to particular types of users. For example, there may be message triples providing bibliographic references, which children would probably find uninteresting.

NaturalOWL provides mechanisms allowing the domain author to assign an importance score to every possible message triple, and possibly different scores for different user types (e.g., adults, children). The score is a non-negative integer indicating how interesting a user of the corresponding type will presumably find the information of the message triple, if the information has not already been conveyed to the user. In the museum projects NaturalOWL was originally developed for, the interest scores ranged from 0 (completely uninteresting) to 3 (very interesting), but a different range could be used. The scores can be specified for all the message triples that involve a particular property P (e.g., $P =$ `madeFrom`), or for all the message triples that involve semantic subjects S of a particular class (e.g., $S \in$ `Statue` or $S =$ `Statue`) and a particular property P, or for message triples that involve particular semantic subjects (e.g., $S =$`exhibit37`) and a particular property P. For example, we may wish to specify that the materials of the exhibits in a collection are generally of medium interest ($P =$ `madeFrom`, score 2), that the materials of statues are of lower interest ($S \in$ `statue`, $P =$ `madeFrom`, score 1), perhaps because all the statues of the collection are made from stone, but that the material of the particular statue `exhibit24` is very important ($S =$ `exhibit10`, $P =$ `madeFrom`, score 3), perhaps because `exhibit24` is a gold statue.

We do not discuss the mechanisms that can be used to assign interest scores to message triples in this chapter, but a detailed description of these mechanisms can be found elsewhere (Androutsopoulos et al., 2012). We also note that when human-authored texts describing individuals and classes of the ontology are available along with the OWL statements or, more generally, the logical facts they express, statistical and machine learning methods can be employed to learn to automatically select or assign interest scores to logical facts (Duboue and McKeown, 2003; Barzilay and Lapata, 2005; Kelly et al., 2009). Another possibility (Demir et al., 2010) would be to compute the interest scores with graph algorithms like PageRank (Brin and Page, 1998).

The domain author can also specify how many times each message triple has to be repeated, before it can be assumed that users of different types have assimilated it. Once a triple has been assimilated, it is never repeated in texts for the same user. For example, the domain author can specify that children assimilate the duration of a historical period when it has been mentioned twice; hence, the system may repeat, for example, the duration of the archaic period in two texts. NaturalOWL maintains a personal model for each end-user. The model shows which message triples were conveyed to the particular user in previous texts, and how many times. More information about the user modelling mechanisms of NaturalOWL can be found elsewhere (Androutsopoulos et al., 2012).

Selecting the message triples to convey

When asked to describe a target, NaturalOWL first retrieves from the ontology the relevant OWL statements, possibly also for second-level targets. It then converts the retrieved statements to message triples, and consults their interest scores and the personal user models to rank the message triples by decreasing interest score, discarding triples that have already been assimilated. If a message triple about the target has been assimilated, then all the message triples about second-level targets that are connected to the assimilated triple are also discarded; for example, if the `creationPeriod` triple (edge) of Figure 2.2 has been assimilated, then the triples about the archaic period

(the edges leaving from `archaicPeriod`) are also discarded. The system then selects up to `maxMessagesPerPage` triples from the most interesting remaining ones; `maxMessagesPerPage` is a parameter whose value can be set to smaller or larger values for types of users that prefer shorter or longer texts, respectively.

Limitations of content selection

OWL allows one to define the broadest possible domain and range of a particular property, using statements like the following.

```
ObjectPropertyDomain(:madeFrom :Wine)  ObjectPropertyRange(:madeFrom :Grape)
```

In practice, more specific range restrictions are then imposed for particular subclasses of the property's domain. For example, the following statements specify that when `madeFrom` is used with individuals from the subclass `GreekWine` of `Wine`, the range (possible values) of `madeFrom` should be restricted to individuals from the subclass `GreekGrape` of `Grape`.

```
SubClassOf(:GreekWine :Wine)  SubClassOf(:GreekGrape :Grape)
SubClassOf(:GreekWine AllValuesFrom(:madeFrom :GreekGrape))
```

NaturalOWL considers `AllValuesFrom` and similar restrictions (see Tables 2.1 and 2.2), but not `ObjectPropertyDomain` and `ObjectPropertyRange` statements. The latter typically provide too general and, hence, uninteresting information from the perspective of end-users.

More generally, NaturalOWL does not consider OWL statements that express axioms about properties, meaning statements declaring that a property is symmetric, asymmetric, reflexive, irreflexive, transitive, functional, that its inverse is functional, that a property is the inverse of, or disjoint with another property, that it is subsumed by a chain of other properties, or that it is a subproperty (more specific) of another property. Statements of this kind are mostly useful in consistency checks, in deduction, or when

generating texts describing the properties themselves (e.g., what being a grandparent of somebody means).[15]

2.3.1.2 Text Planning

For each target, the previous mechanisms produce the message triples to be expressed, with each triple intended to be easily expressible as a single sentence. The text planner of NaturalOWL then orders the triples, in effect ordering the corresponding sentences.

Global and local coherence

When considering *global coherence*, text planners attempt to build a structure, usually a tree, that shows how the clauses, sentences, or larger segments of the text are related to each other, often in terms of rhetorical relations (Mann and Thompson, 1998). The allowed or preferred orderings of the sentences (or segments) often follow, at least partially, from the global coherence structure. In the texts, however, that NaturalOWL is intended to generate, the global coherence structures tend to be rather uninteresting, because most of the sentences simply provide additional information about the target or the second-level targets, which is why global coherence is not considered in NaturalOWL.[16]

When considering *local coherence*, text planners usually aim to maximize measures that examine whether or not adjacent sentences (or segments) continue to focus on the same entities or, if the focus changes, how smooth the transition is. Many local coherence measures are based on Centering Theory (CT) (Grosz et al., 1995; Poesio et al., 2004). Consult the work of Karamanis et al. (2009) for an introduction to CT and a CT-based analysis M-PIRO's texts, which also applies to the texts of NaturalOWL.

[15] Subproperties without sentence plans, discussed below, could inherit sentence plans from their superproperties, but in that case we automatically extract sentence plans from the ontology instead.

[16] Liang et al. (2011a) and Power (2011) seem to agree that very few rhetorical relations are relevant when generating texts from OWL ontologies.

When the maximum fact distance of NaturalOWL is 1, all the sentence-to-sentence transitions are of a type known in CT as CONTINUE, which is the preferred type. If the maximum fact distance is 2, however, the transitions are not always CONTINUE. We repeat below the long aryballos description of page 24 without sentence aggregation. For readers familiar with CT, we show in italics the most salient noun phrase of each sentence u_n, which realizes the discourse entity known as the *preferred center* $c_p(u_n)$. The underlined noun phrases realize the *backward looking center* $c_b(u_n)$, roughly speaking the most salient discourse entity of the previous sentence that is also mentioned in the current sentence.

(1) *This* (exhibit) is an aryballos. (2) *An aryballos* is a kind of vase. (3) *An aryballos* was a small spherical vase with a narrow neck, in which the athletes kept the oil they spread their bodies with. • (4) *This aryballos* was found at the Heraion of Delos. (5) *It* was created during the archaic period. (6) *The archaic period* was when the Greek ancient city-states developed. (7) *It* spans from 700 BC to 480 BC. • (8) *This aryballos* was decorated with the black-figure technique. (9) In *the black-figure technique*, the silhouettes are rendered in black on the pale surface of the clay, and details are engraved. • (10) *This aryballos* is currently in the Museum of Delos.

In sentence 4, where $c_p(u_4)$ is the target exhibit, $c_b(u_4)$ is undefined and the transition from sentence 3 to 4 is a NOCB, a type of transition to be avoided; we mark NOCB transitions with bullets. In sentence 6, $c_p(u_6) = c_b(u_6) \neq c_b(u_5)$, and we have a kind of transition known as SMOOTH-SHIFT (Poesio et al., 2004), less preferred than CONTINUE, but better than NOCB. Another NOCB occurs from sentence 7 to 8, followed by a SMOOTH-SHIFT from sentence 8 to 9, and another NOCB from sentence 9 to 10. All the other transitions are CONTINUE.

The text planner of NaturalOWL groups together sentences (message triples) that describe a particular second-level target (e.g., sentences 2–3, 6–7, and 9) and places each group immediately after the sentence that introduces the corresponding second-level

target (immediately after sentences 1, 5, and 8). Thus the transition from a sentence that introduces a second-level target to the first sentence that describes the second-level target (e.g., from sentence 1 to 2, from 5 to 6, from 8 to 9) is a SMOOTH-SHIFT (or a CONTINUE in the special case from the initial sentence 1 to 2). A NOCB occurs only at sentences that return to providing information about the primary target, after a group of sentences that provide information about a second-level target. All the other transitions are of type CONTINUE.

A simple strategy to avoid NOCB transitions would be to end the generated text once all the message triples that describe a second-level target have been reported, and record in the user model that the other message triples that content selection provided were not actually conveyed. In our example, this would generate sentences 1 to 3; then if the user requested more information about the exhibit, sentences 4 to 7 would be generated etc.

Topical order

When ordering sentences, we also need to consider the topical similarity of adjacent sentences. Compare, for example, the following two texts.

{$_{locationSection}$ The Stoa of Zeus Eleutherios is located in the western part of the Agora. It is located next to the Temple of Apollo Patroos.} {$_{buildSection}$ It was built around 430 BC. It was built in the Doric style. It was built out of porous stone and marble.} {$_{useSection}$ It was used during the Classical period, the Hellenistic period, and the Roman period. It was used as a religious place and a meeting point.} {$_{conditionSection}$ It was destroyed in the late Roman period. It was excavated in 1891 and 1931. Today it is in good condition.}

The Stoa of Zeus Eleutherios was built in the Doric style. It was excavated in 1891 and 1931. It was built out of porous stone and marble. It is located in the western part of the Agora. It was destroyed in the late Roman period. It was used as a religious place and a meeting point. It is located next to the Temple of Apollo Patroos. It was built around 430 BC. Today it is in good condition. It was used during the Classical period, the Hellenistic period, and the Roman period.

Even though both texts contain the same sentences, the second text is more difficult to follow, if at all acceptable. The first one is better, because it groups together topically related sentences. We mark the sentence groups in the first text by curly brackets, but the brackets would not be shown to end-users. In longer texts, sentence groups may optionally be shown as separate paragraphs or sections, which is why we call them *sections*.

For the message triples (and the corresponding sentences) to be grouped by topic, the domain author may define sections (e.g., `locationSection`, `buildSection`) and assign each property to a single section (e.g., assign the properties `isInArea` and `isNextTo` to `locationSection`). Each message triple is then placed in the section of its property. An ordering of the sections and of the properties inside each section can also be specified, causing the message triples to be ordered accordingly (e.g., we may specify that `locationSection` should precede `buildSection`, and that inside `locationSection`, the `isInArea` property should be expressed before `isNextTo`). The sections, the assignments of the properties to sections, and the order of the sections and the properties are defined in the domain-dependent generation resources (Androutsopoulos et al., 2012) using the Protégé plug-in of NaturalOWL.

The overall text planning algorithm

NaturalOWL's text planning algorithm is summarized in Figure 2.3. If the message triples to be ordered include triples that describe second-level targets, i.e., triples $\langle S, P, O \rangle$ whose owner S is a second-level target, then the triples of the primary and each second-level target are ordered separately, using the ordering of properties and sections. The ordered triples of each second-level target are then inserted into the ordered list of the primary target triples, immediately after the first triple that introduces the second-level target, i.e., immediately after the first triple whose O is the second-level target.

Further related work on text planning

The ordering of properties and sections is similar to *text schemata* (McKeown,

```
procedure orderMessageTriples
inputs:
  t[0]: primary target
  t[1], ..., t[n]: second-level targets
  L[0]: unordered list of triples describing t[0]
  ...
  L[n]: unordered list of triples describing t[n]
  SMap: mapping from properties to sections
  SOrder: partial order of sections
  POrder: partial order of properties within sections
output:
  ordered list of message triples
steps:
  for i := 0 to n { orderMessageTriplesAux(L[i], SMap, SOrder, POrder) }
  for i := 1 to n { insertAfterFirst(<t[0], _, t[i]>, L[0], L[i]) }
  return L[0]

procedure orderMessageTriplesAux
inputs:
  L: unordered list of triples about a single target
  SMap: mapping from properties to sections
  SOrder: partial order of sections
  POrder: partial order of properties within sections
local variables:
  S[1], ..., S[k]: lists, each with triples of one section
output:
  ordered list of message triples about a single target
steps:
  <S[1], ..., S[k]> := splitInSections(L, SMap)
  for i := 1 to k { S[i] := orderTriplesInSection(S[i], POrder) }
  <S[1], ..., S[k]> := reorderSections(S[1], ..., S[k], SOrder)
  return concatenate(S[1], ..., S[k])
```

Figure 2.3: The overall text planning algorithm of NaturalOWL.

1985), roughly speaking domain-dependent patterns that specify the possible arrangements of different types of sentences (or segments). Sentence ordering has been studied extensively in text summarization (Barzilay et al., 2002). Duboue and McKeown (2001) discuss methods that could be used to learn the order of sentences or other segments in NLG from semantically tagged training corpora. Consult also the work of Barzilay and Lee (2004), Elsner et al. (2007), Barzilay and Lapata (2008), and Chen et al. (2009).

2.3.2 Micro-planning

The processing stages we have discussed so far select and order the message triples to be expressed. The next stage, *micro-planning*, consists of three sub-stages: *lexicalization*, *sentence aggregation*, and *generation of referring expressions*; see also Figure 2.1.

2.3.2.1 Lexicalization

During lexicalization, NLG systems usually turn the output of content selection (in our case, the message triples) to abstract sentence specifications. In NaturalOWL, for every property of the ontology and every supported natural language, the domain author may specify one or more template-like sentence plans to indicate how message triples involving that property can be expressed. We discuss below how sentence plans are specified, but first a slight deviation is necessary, to briefly discuss the lexicon entries of NaturalOWL.

Lexicon entries

For each verb, noun, or adjective that the domain author wishes to use in the sentence plans, a lexicon entry has to be provided, to specify the inflectional forms of that word.[17] All the lexicon entries are multilingual (currently bilingual); this allows sentence plans to be reused across similar languages when no better option is available,

[17] No lexicon entries need to be provided for closed-class words, like determiners and prepositions.

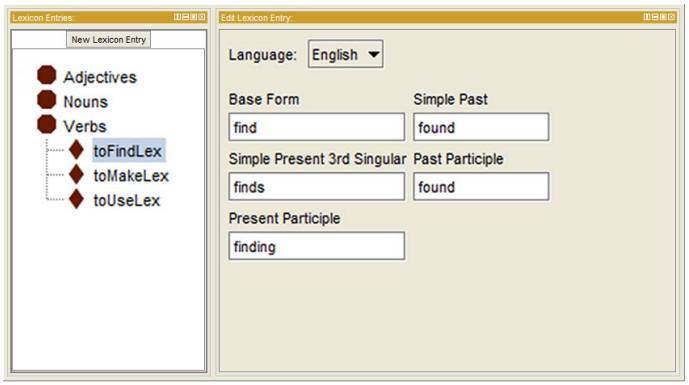


Figure 2.4: A lexicon entry for the verb “to find”.

as discussed elsewhere (Androutsopoulos et al., 2007). Figure 2.4 shows the lexicon entry for the verb whose English base form is “find”, as viewed by the domain author when using the Protégé plug-in of NaturalOWL. The identifier of the lexicon entry is `toFindLex`. The English part of the entry shows that the base form is “find”, the simple past is “found” etc. Similarly, the Greek part of the lexicon entry would show the base form of the corresponding verb (“βρίσκω”) and its inflectional forms in the various tenses, persons etc. The lexicon entries for nouns and adjectives are very similar.

Most of the English inflectional forms could be automatically produced from the base forms by using simple morphology rules. We hope to exploit an existing English morphology component, such as that of SIMPLENLG (Gatt and Reiter, 2009), in future work. Similar morphology rules for Greek were used in the authoring tool of M-PIRO (Androutsopoulos et al., 2007), and we hope to include them in a future version of NaturalOWL. Rules of this kind would reduce the time a domain author spends creating lexicon entries. In the ontologies we have considered, however, a few dozens of lexicon entries for verbs, nouns, and adjectives suffice. Hence, even without facilities to automatically produce inflectional forms, creating the lexicon entries is rather trivial. Another possibility would be to exploit a general-purpose lexicon or lexical database, like WordNet (Fellbaum, 1998) or CELEX, though resources of this kind often do not

cover the highly technical concepts of ontologies.[18]

The lexicon entries and, more generally, all the domain-dependent generation resources of NaturalOWL are stored as instances of an OWL ontology (other than the ontology the texts are generated from) that describes the linguistic resources of the system (Androutsopoulos et al., 2012). The domain author, however, interacts with the plug-in and does not need to be aware of the OWL representation of the resources. By representing the domain-dependent generation resources in OWL, it becomes easier to publish them on the Web, check for inconsistencies etc., as with other OWL ontologies.

Sentence plans

In NaturalOWL, a sentence plan is a sequence of slots, along with instructions specifying how to fill them in. Figure 2.5 shows an English sentence plan for the property `usedDuringPeriod`, as viewed by the domain author when using the Protégé plug-in of NaturalOWL. The sentence plan expresses message triples of the form $\langle S, : usedDuringPeriod, O \rangle$ by producing sentences like the following.

[$_{slot_1}$This stoa] [$_{slot_2}$was used] [$_{slot_3}$during] [$_{slot_4}$the Classical period].

[$_{slot_1}$The Stoa of Zeus Eleutherios] [$_{slot_2}$was used] [$_{slot_3}$during] [$_{slot_4}$the Classical period, the Hellenistic period, and the Roman period].

The first slot of the sentence plan of Figure 2.5 is to be filled in with an automatically generated referring expression for the owner (S) of the triple. For example, if the triple to express is `<:stoaZeusEleutherios, :usedDuringPeriod, :classicalPeriod>`, an appropriate referring expression for S may be a demonstrative noun phrase like "this stoa", a pronoun ("it"), or the monument's natural language name ("the Stoa of Zeus Eleutherios"). We discuss the generation of referring expressions below, along with mechanisms to specify natural language names. The sentence

[18]For more information on CELEX, please consult `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC96L14`.

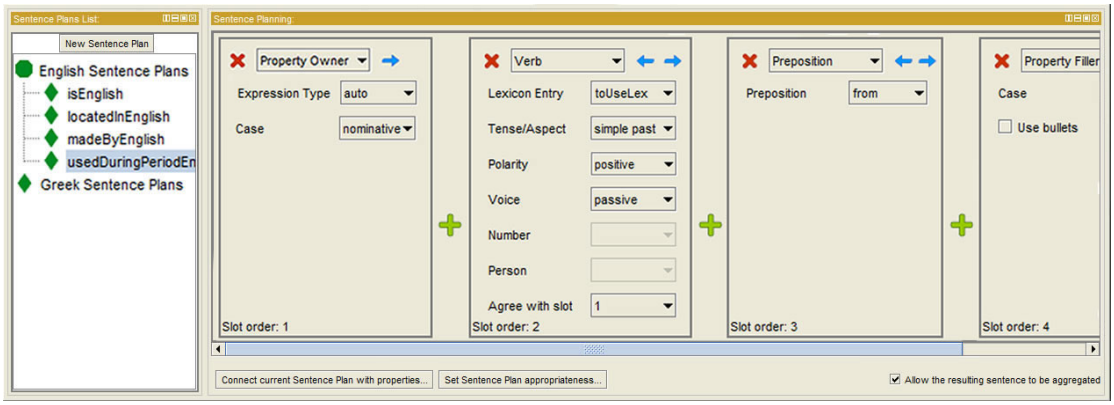


Figure 2.5: A sentence plan for the property `usedDuringPeriod`.

plan also specifies that the referring expression must be in nominative case (e.g., "it" or "this stoa", as opposed to the genitive case expressions "its" or "this stoa's", as in "This stoa's height is 5 meters").

The second slot is to be filled in with a form of the verb whose lexicon identifier is `toUseVerb`. The verb form must be in the simple past and passive voice, in positive polarity (as opposed to "was *not* used"). Its number must agree with the number of the expression in the first slot; for example, we want to generate "The Stoa of Zeus Eleutherios *was* used", but "Stoas *were* used". The third slot is filled in with the preposition "during". The fourth slot is filled in with an expression for the filler (O) of the message triple, in accusative case.[19] With `<:stoaZeusEleutherios, :usedDuringPeriod, :classicalPeriod>`, the slot would be filled in with the natural language name of `classicalPeriod`.[20] The sentence plan also allows the resulting sentence to be aggregated with other sentences.

More generally, the instructions of a sentence plan may indicate that a slot should be filled in with one of the following (i–vii):

[19]English prepositions usually require noun phrase complements in accusative (e.g., "on him"). In Greek and other languages, cases have more noticeable effects.

[20]Future versions of NaturalOWL may allow a referring expression for O other than its natural language name to be produced (e.g., a pronoun), as with S.

(i) *A referring expression for the S* (owner) of the message triple. A sentence plan may specify a particular type of referring expression to use (e.g., always use the natural language name of *S*) or, as in the example of Figure 2.5, it may allow the system to automatically produce the most appropriate type of referring expression depending on the context.

(ii) *A verb* for which there is a lexicon entry, in a particular form, possibly a form that agrees with another slot. The polarity of the verb can also be manually specified or, if the filler (*O*) of the message triple is a Boolean value, the polarity can be automatically set to match it (e.g., to produce "It does *not* have a built-in flash" when *O* is `false`).

(iii) *A noun or adjective* from the lexicon, in a particular form (e.g., case, number), or in a form that agrees with another slot.

(iv) *A preposition* or (v) *a fixed string*.

(vi) *An expression for the O* (filler) of the triple. If *O* is an individual or class, then the expression is the natural language name of *O*; if *O* is a datatype value (e.g., an integer), then the value itself is inserted in the slot; and similarly if *O* is a disjunction or conjunction of datatype values, individuals, or classes.

(vii) *A concatenation of property values of O*, provided that *O* is an individual. For example, we may need to express a message triple like the first one below, whose (anonymous in the RDF sense) object `_:n` is linked to both a numeric value (via the property `hasAmount`) and an individual standing for the currency (via `hasCurrency`).

```
<:tecra8, :hasPrice, _:n>
<_:n, :hasAmount, "850"^^xsd:float>
<_:n, :hasCurrency, :euroCurrency>
```

We would want the sentence plan to include a slot filled in with the concatenation of the `hasAmount` value of `_:n` and the natural language name of the `hasCurrency` value of `_:n` (e.g., "850 Euro" in English, "850 Ευρώ" in Greek).

Default sentence plan

If no sentence plan has been provided for a particular property of the ontology, NaturalOWL uses a default sentence plan, consisting of three slots. The first slot is filled in with an automatically generated referring expression for the owner (*S*) of the triple, in nominative case. The second slot is filled in with a tokenized form of the OWL identifier of the property. The third slot is filled in with an appropriate expression for the filler (*O*) of the triple, as discussed above, in accusative case (if applicable). For the following message triple, the default sentence plan would produce the sentence shown below:

```
<:stoaZeusEleutherios, :usedDuringPeriod,
   and(:classicalPeriod, :hellenisticPeriod, :romanPeriod)>
```

Stoa zeus eleutherios used during period classical period, hellenistic period, and roman period.

Notice that we use a single message triple with an `and(...)` filler, instead of a different triple for each period. This kind of triple merging is in effect a form of aggregation, discussed below, but it takes place during content selection. Also, we assumed in the sentence above that the natural language names of the individuals have not been provided either; in this case, NaturalOWL uses tokenized forms of the OWL identifiers of the individuals instead. The tokenizer of NaturalOWL can handle both CamelCase (e.g., `usedDuringPeriod`) and underscore style (e.g., `used_during_period`). When other styles are used in the identifiers of properties, classes, and individuals, the output of the tokenizer may be worse than the example suggests, but the resulting sentences can be improved by providing sentence plans and by associating classes and individuals with natural language names, discussed below.

Using rdfs:label strings

OWL properties (and other elements of OWL ontologies) can be labeled with strings in multiple natural languages using the `rdfs:label` annotation property, defined in the RDF and OWL standards. For example, the `usedDuringPeriod` property could be labeled with "was used during" as shown below; there could be similar labels for Greek and other languages.

```
AnnotationAssertion(rdfs:label :usedDuringPeriod "was used during"@en)
```

If an `rdfs:label` string has been specified for the property of a message triple, NaturalOWL uses that string in the second slot of the default sentence plan. The quality of the resulting sentences can, thus, be improved, if the `rdfs:label` strings are more natural phrases than the tokenized property identifiers. With the `rdfs:label` shown above, the default sentence plan would produce the following sentence.

Stoa zeus eleutherios was used during classical period, hellenistic period, and roman period.

Even with `rdfs:label` strings, the default sentence plan may produce sentences with disfluencies. Also, the `rdfs:label` strings do not indicate the grammatical categories of their words, and this does not allow the system to apply many of the sentence aggregation rules discussed below. A further limitation of the default sentence plan is that it does not allow the slots for *S* and *O* to be preceded or followed, respectively, by any other phrase.

Sentence plans for domain-independent and modified properties

The domain author does not need to provide sentence plans for domain-independent properties (e.g., `instanceOf`, `isA`, see Tables 2.1–2.2). These properties have fixed, domain-independent semantics; hence, built-in sentence plans are used. The English built-in sentence plans, which also serve as further examples of sentence plans, are summarized in Table 2.3; the Greek built-in sentence plans are similar. To save space we show the sentence plans as templates in Table 2.3, and we do not show the sentence plans for negated domain-independent properties (e.g., `not(isA)`), which are similar. Additional slot restrictions not shown in Figure 2.3 require, for example, subject-verb number agreement and the verb forms ("is" or "was") to be in present tense. Information provided when specifying the natural language names of individuals and classes, discussed below, shows if definite or indefinite articles or no articles at all should be used (e.g., "*the* N97 mini", "exhibit 24", "*a* St. Emilion" or "*the* St. Emilion" or simply

Forms of message triples and the corresponding built-in sentence plans	Example message triples and possible resulting sentences
<*S*, `instanceOf`, *O*> *ref*(*S*) `toBeVerb` *name*(*indef*, *O*)	`<:eos450d, instanceOf, :PhotographicCamera>` The EOS 450D is a photographic camera.
<*S*, `instanceOf`, *O*> *ref*(*S*) `toBeVerb` *name*(*adj*, *O*)	`<:eos450d, instanceOf, :Cheap>` The EOS 450D is cheap.
<*S*, `oneOf`, *O*> *ref*(*S*) `toBeVerb` *name*(*O*)	`<:WineColor, oneOf, or(:white, :rose, :red)>` A wine color is white, rose, or red.
<*S*, `differentIndividuals`, *O*> *ref*(*S*) `toBeVerb` not identical to *name*(*O*)	`<:n97, differentIndividuals, :n97mini>` The N97 is not identical to the N97 mini.
<*S*, `sameIndividual`, *O*> *ref*(*S*) `toBeVerb` identical to *name*(*O*)	`<:eos450d, sameIndividual, :rebelXSi>` It is identical to the Rebel XSi.
<*S*, `isA`, *O*> *ref*(*S*) `toBeVerb` a kind of *name*(*noarticle*, *O*)	`<:StEmilion, isa, :Bordeaux>` St. Emilion is a kind of Bordeaux.
<*S*, `isA`, *O*> *ref*(*S*) `toBeVerb` *name*(*adj*, *O*)	`<:StEmilion, isa, :Red>` St. Emilion is red.
Notation: *ref*(ξ) stands for a referring expression for ξ; *name*(ξ) is the natural language name of ξ; *name*(*indef*, ξ) and *name*(*noarticle*, ξ) mean that the name should be a noun phrase with an indefinite or no article. Sentence plans involving *name*(*adj*, ξ) are used when the natural language name of ξ is a sequence of one or more adjectives; otherwise the sentence plan of the previous row is used.	

Table 2.3: Built-in English sentence plans for domain-independent properties.

"St. Emilion"), and what the default number of each name is (e.g., "A wine color is" or "Wine colors are"). It is also possible to modify the built-in sentence plans; for example, in a museum context we may wish to generate "An aryballos *was* a kind of vase" instead of "An aryballos *is* a kind of vase".

For modified properties (e.g., `minCardinality(manufacturedBy)`, see Tables 2.1–2.2) the sentence plans are also automatically produced, from the sentence plans of the unmodified properties (e.g., `manufacturedBy`).

Specifying the appropriateness of sentence plans

Multiple sentence plans may be provided for the same property of the ontology and

the same language. Different appropriateness scores (similar to the interest scores of properties) can then be assigned to alternative sentence plans per user type. This allows specifying, for example, that a sentence plan that generates sentences like "This amphora depicts Miltiades" is less appropriate when interacting with children, compared to an alternative sentence plan with a more common verb (e.g., "shows"). Automatically constructed sentence plans inherit the appropriateness scores of the sentence plans they are constructed from.

Related work on sentence plans

The sentence plans of NaturalOWL are similar to expressions of sentence planning languages like SPL (Kasper and Whitney, 1989) that are used in generic surface realizers, such as FUF/SURGE (Elhadad and Robin, 1996), KPML (Bateman, 1997), REALPRO (Lavoie and Rambow, 1997), NITROGEN/HALOGEN (Langkilde, 2000), and OPENCCG (White, 2006). The sentence plans of NaturalOWL, however, leave fewer decisions to subsequent stages. This has the disadvantage that our sentence plans often include information that could be obtained from large-scale grammars or corpora (Wan et al., 2010). On the other hand, the input to generic surface realizers often refers to non-elementary linguistic concepts (e.g., features of a particular syntax theory) and concepts of an *upper model* (Bateman, 1990); the latter is a high-level domain-independent ontology that may use a very different conceptualization than the ontology the texts are to be generated from. Hence, linguistic expertise, for example in Systemic Grammars (Halliday, 1994) in the case of KPML (Bateman, 1997), and effort to understand the upper model are required. By contrast, the sentence plans of NaturalOWL require the domain author to be familiar with only elementary linguistic concepts (e.g., tense, number), and they do not require familiarity with an upper model. Our sentence plans are simpler than, for example, the templates of Busemann and Horacek (1998) or McRoy et al. (2003), in that they do not allow, for instance, conditionals or recursive invocation of other templates. See also the work of Reiter (1995) and van Deemter et al. (2005) for a discussion

of template-based vs. more principled NLG.

When corpora of texts annotated with the message triples they express are available, templates can also be automatically extracted (Ratnaparkhi, 2000; Angeli et al., 2010; Duma and Klein, 2013). We discuss methods of this kind in Chapter 3, along with methods to automatically or semi-automatically extract the domain-dependent generation resources of NaturalOWL from the Web. Statistical methods that jointly perform content selection, lexicalization, and surface realization have also been proposed (Liang et al., 2009; Konstas and Lapata, 2012a; Konstas and Lapata, 2012b), but they are currently limited to generating single sentences from flat records. We discuss methods of this kind in Chapter 4, along with our Integer Linear Programming model to jointly perform content selection, lexicalization, and sentence aggregation.

Specifying natural language names

The domain author can assign *natural language* (NL) names to the individuals and named classes of the ontology; recall that by named classes we mean classes that have OWL identifiers. If an individual or named class is not assigned an NL name, then its `rdfs:label` or a tokenized form of its identifier is used instead. The NL names that the domain author provides are specified much as sentence plans, i.e., as sequences of slots. For example, we may specify that the English NL name of the class `ItalianWinePiemonte` is the concatenation of the following slots; we explain the slots below.

[*indef* an] [*adj* Italian] [*headnoun* wine] [*prep* from] [*def* the] [*noun* Piemonte] [*noun* region]

This would allow NaturalOWL to generate the sentence shown below from the following message triple; a tokenized form of the identifier of `wine32` is used.

```
<:wine32, instanceOf, :ItalianWinePiemonte>
```

Wine 32 is an Italian wine from the Piemonte region.

Similarly, we may assign the following NL names to individuals `classicalPeriod`, `stoaZeusEleutherios`, `gl2011`, and classes `ComputerScreen` and `Red`. NaturalOWL makes no distinction between common and proper nouns; both are entered as nouns in the lexicon, and may be multi-word (e.g., "Zeus Eleutherios"). NaturalOWL can also be instructed to capitalize the words of particular slots (e.g., "Classical").

[*def* the] [*adj* Classical] [*headnoun* period] , [*def* the] [*headnoun*stoa] [*prep*of] [*noun*Zeus Eleutherios],

[*headnoun* GL-2011] , [*indef* a] [*noun* computer] [*headnoun* screen] , [*headadj* red]

These NL names could be used to express the message triples shown below:

```
<:stoaZeusEleutherios, :usedDuringPeriod,  :classicalPeriod>
```

The Stoa of Zeus Eleutherios was used during the Classical period.

```
<:gl2011, instanceOf, :ComputerScreen>  <:gl2011, instanceOf, :Red>
```

GL-2011 is a computer screen. GL-2011 is red.

More precisely, each NL name is a sequence of slots, with accompanying instructions specifying how the slots are to be filled in. Each slot can be filled in with:

(i) *An article*, definite or indefinite. The article in the first slot (if present) is treated as the article of the overall NL name.

(ii) *A noun or adjective flagged as the head* (main word) of the NL name. Exactly one head must be specified per NL name and it must have a lexicon entry. The number and case of the head, which is also taken to be the number and case of the overall NL name, can be automatically adjusted per context. For example, different sentence plans may require the same NL name to be in nominative case when used as a subject, but in accusative when used as the object of a verb; and some aggregation rules, discussed below, may require a singular NL name to be turned into plural. Using the lexicon entries, which list the inflectional forms of nouns and adjectives, NaturalOWL can adjust the NL

names accordingly. The gender of head adjectives can also be automatically adjusted, whereas the gender of head nouns is fixed and specified by their lexicon entries.

(iii) *Any other noun or adjective*, among those listed in the lexicon. The NL name may require a particular inflectional form to be used, or it may require an inflectional form that agrees with another slot of the NL name.

(iv) *A preposition*, or (v) *any fixed string*.

As with sentence plans, the domain author specifies NL names by using the Protégé plug-in of NaturalOWL. Multiple NL names can be specified for the same individual or class, and they can be assigned different appropriateness scores per user type; hence, different terminology (e.g., common names of diseases) can be used when generating texts for non-experts, as opposed to texts for experts (e.g., doctors). The domain author can also specify, again using the plug-in, if the NL names of particular individuals or classes should involve definite, indefinite, or no articles, and if the NL names should be in singular or plural by default. For example, we may prefer the texts to mention the class of aryballoi as a single particular generic object, or by using an indefinite singular or plural form, as shown below.

The aryballos is a kind of vase. An aryballos is a kind of vase. Aryballoi are a kind of vase.

2.3.2.2 Sentence Aggregation

The sentence plans of the previous section lead to a separate sentence for each message triple. NLG systems often aggregate sentences into longer ones to improve readability. In NaturalOWL, the maximum number of sentences that can be aggregated to form a single longer sentence is specified per user type via the `maxMessagesPerSentence` parameter. In the museum contexts our system was originally developed for, setting `maxMessagesPerSentence` to 3 or 4 led to reasonable texts for adult visitors, whereas a value of 2 was used for children. The sentence aggregation of NaturalOWL is performed by a set of manually crafted rules, intended to be domain-independent. We do

not claim that this set of rules, which was initially based on the aggregation rules of M-PIRO (Melengoglou, 2002), is complete, and we hope it will be extended in future work; see, for example, the work of Dalianis (1999) for a rich set of aggregation rules.[21] Nevertheless, the current rules of NaturalOWL already illustrate several aggregation opportunities that arise when generating texts from OWL ontologies.

To save space, we discuss only English sentence aggregation; Greek aggregation is similar. We show mostly example *sentences* before and after aggregation, but the rules actually operate on *sentence plans* and they also consider the message triples being expressed. The rules are intended to aggregate short single-clause sentences. Sentence plans that produce more complicated sentences may be flagged (using the tickbox at the bottom of Figure 2.5) to signal that aggregation should not affect their sentences. The aggregation rules apply almost exclusively to sentences that are adjacent in the ordering produced by the text planner; the only exception are aggregation rules that involve sentences about cardinality restrictions. Hence, depending on the ordering of the text planner there may be more or fewer aggregation opportunities; see the work of Cheng and Mellish (2000) for related discussion. Also, the aggregation rules of NaturalOWL operate on sentences of the same topical section, because aggregating topically unrelated sentences often sounds unnatural.

The aggregation of NaturalOWL is greedy. For each of the rules discussed below, starting from those discussed first, the system scans the original (ordered) sentences from first to last, applying the rule wherever possible, provided that the rule's application does not lead to a sentence expressing more than `maxMessagesPerSentence` original sentences. If a rule can be applied in multiple ways, for example to aggregate two or three sentences, the application that aggregates the most sentences without violating `maxMessagesPerSentence` is preferred.

[21] When appropriate corpora are available, it may also be possible to train aggregation modules (Walker et al., 2001; Barzilay and Lapata, 2006).

Avoid repeating a noun with multiple adjectives: Message triples of the form $\langle S, P, O_1 \rangle$, ..., $\langle S, P, O_n \rangle$ will have been aggregated into a single message triple of the form $< S, P,$`and(`$O_1, \ldots, O_n$`)`$>$. If the NL names of $O_1, \ldots, O_n$ are, apart from possible initial determiners, sequences of adjectives followed by the same head noun, then the head noun does not need to be repeated. Let us consider the following message triple. Assuming that the NL names of the three periods are as in the first sentence below, the original sentence will repeat "period" three times. The aggregation rule omits all but the last occurrence of the head noun.

```
<:stoaZeusEleutherios, :usedDuringPeriod,
    and(:classicalPeriod, :hellenisticPeriod, :romanPeriod)>
```

It was used during the Classical period, the Hellenistic period, and the Roman period. $\Rightarrow$ It was used during the Classical, the Hellenistic, and the Roman period.

Cardinality restrictions and values: This is a set of rules that aggregate *all* the sentences (not necessarily adjacent) that express message triples of the form $\langle S, M(P), O \rangle$ and $\langle S, P, O \rangle$, for the same S and P, with M being a modifier of either `minCardinality`, `maxCardinality`, or `exactCardinality`. When these rules are applied, the value of `MaxMessagesPerSentence` is ignored. For example, these rules perform aggregations like the following.

Model 35 is sold in at most three countries. Model 35 is sold in at least three countries. Model 35 is sold in Spain, Italy, and Greece. $\Rightarrow$ Model 35 is sold in exactly three countries: Spain, Italy, and Greece.

Class and passive sentence: This rule aggregates (i) a sentence expressing a message triple $\langle S, : instanceOf, C \rangle$ or $\langle S, : isA, C \rangle$ and (ii) a passive immediately subsequent sentence expressing a single triple of the form $\langle S, P, O \rangle$, for the same S, where P is an (unmodified) property of the ontology. The subject and auxiliary verb of the second sentence are omitted.

Bancroft Chardonnay is a kind of Chardonnay. It is made in Bancroft. ⇒ Bancroft Chardonnay is a kind of Chardonnay made in Bancroft.

Class and prepositional phrase: The second sentence now involves the verb "to be" in the active simple present, immediately followed by a preposition; the other conditions are as in the previous rule. The subject and verb of the second sentence are omitted.

Bancroft Chardonnay is a kind of Chardonnay. It is from Bancroft. ⇒ Bancroft Chardonnay is a kind of Chardonnay from Bancroft.

Class and multiple adjectives: This rule aggregates (i) a sentence of the same form as in the previous two rules, and (ii) one or more immediately preceding or subsequent sentences, each expressing a single message triple $\langle S, P_i, O_i \rangle$, for the same S, where P_i are (unmodified) properties of the ontology. Every preceding or subsequent sentences must involve the verb "to be" in the active simple present, immediately followed by only an adjective. The adjectives are absorbed into sentence (i) maintaining their order.

This is a motorbike. It is red. It is expensive. ⇒ This is a red, expensive motorbike.

Same verb conjunction/disjunction: In a sequence of sentences involving the same verb form, each expressing a single message triple $\langle S, P_i, O_i \rangle$, where S is the same in all the triples and P_i are (unmodified) properties of the ontology, a conjunction can be formed by mentioning the subject and verb once. The "and" is omitted if a preposition follows.

It has medium body. It has moderate flavor. ⇒ It has medium body and moderate flavor.

He was born in Athens. He was born in 1918. ⇒ He was born in Athens in 1918.

A similar rule applies to sentences produced from disjunctions of message triples, as illustrated below. A variant of the first aggregation rule is then also applied.

The house wine has strong flavor or it has medium flavor. ⇒ The house wine has strong flavor or medium flavor. ⇒ The house wine has strong or medium flavor.

Different verbs conjunction: If there is a sequence of sentences, not involving the same verb form, each expressing a message triple $\langle S, P_i, O_i \rangle$, where S is the same in all the triples and P_i are (unmodified) properties of the ontology, a conjunction can be formed:

> Bancroft Chardonnay is dry. It has moderate flavor. It comes from Napa. $\Rightarrow$ Bancroft Chardonnay is dry, it has moderate flavor, and it comes from Napa.

2.3.2.3 Generating Referring Expressions

A sentence plan may require a referring expression to be generated for the S of a message triple $\langle S, P, O \rangle$. Depending on the context, it may be better, for example, to use the NL name of S (e.g., "the Stoa of Zeus Eleutherios"), a pronoun (e.g., "it"), a demonstrative noun phrase (e.g., "this stoa") etc. Similar alternatives could be made available for O, but NaturalOWL currently uses O itself, if it is a datatype value; or the NL name of O, its tokenized identifier, or its `rdfs:label`, if O is an entity or class; and similarly for conjunctions and disjunctions in O. Hence, below we focus only on referring expressions for S.

NaturalOWL currently uses a limited range of referring expressions, which includes only NL names (or tokenized identifiers or `rdfs:label` strings), pronouns, and noun phrases involving only a demonstrative and the NL name of a class (e.g., "this vase"). For example, referring expressions that mention properties of S (e.g., "the vase from Rome") are not generated. Although the current referring expression generation mechanisms of NaturalOWL work reasonably well, they are best viewed as placeholders for more elaborate algorithms (Krahmer and van Deemter, 2012), especially algorithms based on description logics (Areces et al., 2008; Ren et al., 2010).

Let us consider the following generated text, which expresses the triples $\langle S_i, P_i, O_i \rangle$ shown below. We do not aggregate sentences in this section, to illustrate more cases where referring expressions are needed; aggregation would reduce, however, the number of pronouns, making the text less repetitive. For readers familiar with CT (Sec-

tion 2.3.1.2), we show again in italics the noun phrase realizing $c_p(u_n)$, we show underlined the noun phrase realizing $c_b(u_n)$, and we mark NOCB transitions with bullets.

(1) *Exhibit 7* is a statue. (2) *It* was sculpted by Nikolaou. (3) *Nikolaou* was born in Athens. (4) *He* was born in 1918. (5) *He* died in 1998. • (6) *Exhibit 7* is now in the National Gallery. (7) *It* is in excellent condition.

```
<:exhibit7, instanceOf, :Statue>
<:exhibit7, :hasSculptor, :nikolaou>
<:nikolaou, :cityBorn, :athens>
<:nikolaou, :yearBorn, "1918"^^xsd:integer>
<:nikolaou, :yearDied, "1998"^^xsd:integer>
<:exhibit7, :currentLocation, :nationalGallery>
<:exhibit7, :currentCondition, :excellentCondition>
```

NaturalOWL pronominalizes S_n (for $n > 1$) only if $S_n = S_{n-1}$, as in sentences 2, 4, 5, and 7. Since typically $c_p(u_i) = S_i$, we obtain $c_p(u_n) = c_p(u_{n-1})$, whenever S_n is pronominalized, if the pronoun is resolved by the reader as intended. People tend to prefer readings where $c_p(u_n) = c_p(u_{n-1})$, if no other restriction is violated (e.g., gender, number, world knowledge). This helps the pronouns that NaturalOWL generates to be correctly resolved by readers, even when they would appear to be potentially ambiguous. For example, the pronoun of sentence 7 is most naturally understood as referring to the exhibit, as it is intended to, not the gallery, even though both are neuter and can be in excellent condition. Note that with both referents, the transition from sentence 6 to 7 is a CONTINUE; hence, transition type preferences play no role. The gender of each generated pronoun is the gender of the (most appropriate) NL name of the S that the pronoun realizes.[22] If S does not have an NL name, NaturalOWL uses the gender of the (most appropriate) NL name of the most specific class that includes S and has an NL name (or one of these classes, if they are many). NL names can also be associated with

[22] In languages like Greek that use grammatical instead of natural genders, the pronouns' genders cannot be determined by consulting the ontology (e.g., to check if the referent is animate or inanimate).

sets of genders, which give rise to pseudo-pronouns like "he/she"; this may be desirable in the NL name of a class like `Person`.

With some individuals or classes, we may not wish to use NL names, nor tokenized identifiers or `rdfs:label` strings. This is common, for example, in museum ontologies, where some exhibits are known by particular names, but many other exhibits are anonymous and their OWL identifiers are not particularly meaningful. NaturalOWL allows the domain author to mark individuals and classes as *anonymous*, to indicate that their NL names, tokenized identifiers, and `rdfs:label` strings should be avoided. When the primary target is marked as anonymous, NaturalOWL uses a demonstrative noun phrase (e.g., "this statue") to refer to it. The demonstrative phrase involves the NL name of the most specific class that subsumes the primary target, has an NL name, and has not been marked as anonymous. Especially in sentences that express `isA` or `instanceOf` message triples about the primary target, the demonstrative phrase is simply "this", to avoid generating sentences like "This statue is a statue". The marking of anonymous individuals and classes currently affects only the referring expressions of the primary target.

2.3.3 Surface Realization

In many NLG systems, the sentences at the end of micro-planning are underspecified; for example, the order of their constituents or the exact forms of their words may be unspecified. Large-scale grammars or statistical models can then be used to fill in the missing information during surface realization, as already discussed (Section 2.3.2.1). By contrast, in NaturalOWL (and most template-based NLG systems) the (ordered and aggregated) sentence plans at the end of micro-planning already completely specify the surface (final) form of each sentence. Hence, the surface realization of NaturalOWL is mostly a process of converting internal, but fully specified and ordered sentence specifications to the final text. Punctuation and capitalization are also added. Application-

specific markup (e.g., HTML tags, hyperlinks) or images can also be added by modifying the surface realization code of NaturalOWL.

2.4 Trials

In previous work, NaturalOWL was used mostly to describe cultural heritage objects. In the XENIOS project, it was tested with an OWL version of an ontology that was created during M-PIRO to document approximately 50 archaeological exhibits (Androutsopoulos et al., 2007).[23] The OWL version comprised 76 classes, 343 individuals (including cities, persons etc.), and 41 properties. In XENIOS, NaturalOWL was also embedded in a robotic avatar that presented the exhibits of M-PIRO in a virtual museum (Oberlander et al., 2008). More recently, in the INDIGO project, NaturalOWL was embedded in mobile robots acting as tour guides in an exhibition about the ancient Agora of Athens.[24] An OWL ontology documenting 43 monuments was used; there were 49 classes, 494 individuals, and 56 properties in total.

In XENIOS and INDIGO, the texts of NaturalOWL were eventually indistinguishable from human-authored texts. We participated, however, in the development of the ontologies, and we may have biased them towards choices (e.g., classes, properties) that made it easier for NaturalOWL to generate high-quality texts. Hence, in the trials discussed below, we wanted to experiment with independently developed ontologies. We also wanted to experiment with different domains, as opposed to cultural heritage.

[23] XENIOS was co-funded by the European Union and the Greek General Secretariat of Research and Technology; see `http://www.ics.forth.gr/xenios/`.

[24] INDIGO was an FP6 IST project of the European Union; for more available information consult `http://www.ics.forth.gr/indigo/`. Videos that show the robots of XENIOS and INDIGO are available at `http://nlp.cs.aueb.gr/projects.html`. Two AUEB students, G. Karakatsiotis and V. Pterneas, won the Interoperability Challenge of the 2011 Microsoft Imagine Cup with a similar mobile phone application, called Touring Machine, which uses NaturalOWL; see `http://www.youtube.com/watch?v=PaNAmNC7dZw`.

A further goal was to compare the texts of NaturalOWL against those of a simpler verbalizer. We used the OWL verbalizer of the SWAT project (Stevens et al., 2011; Williams et al., 2011), which we found to be particularly robust and useful.[25] The verbalizer produces an alphabetical glossary with an entry for each named class, property, and individual, without requiring domain-dependent generation resources. Each glossary entry is a sequence of English-like sentences expressing the corresponding OWL statements of the ontology. The SWAT verbalizer uses a predetermined partial order of statements in each glossary entry; for example, when describing a class, statements about equivalent classes or super-classes are mentioned first, and individuals belonging in the target class are mentioned last.[26] The verbalizer actually translates the OWL ontology to Prolog, it extracts lexicon entries from OWL identifiers and `rdfs:label` strings, and it uses predetermined sentence plans specified as a DCG grammar. It also aggregates, in effect, message triples of the same property that share one argument (*S* or *O*) (Williams and Power, 2010).

Our hypothesis was that the domain-dependent generation resources would help NaturalOWL produce texts that end-users would consider more fluent and coherent, compared to those produced by the SWAT verbalizer, but also those produced by NaturalOWL without domain-dependent generation resources. We also wanted to demonstrate that high-quality texts could be produced in both English and Greek, and to measure the effort required to create the domain-dependent generation resources of NaturalOWL for existing ontologies. This effort had not been measured in previous work,

[25]The SWAT verbalizer can be used on-line at `http://swat.open.ac.uk/tools/`. We used the general-purpose version that was on-line in July and August 2011; a similar verbalizer from OWL to ACE (Section 2.2) is available at `http://attempto.ifi.uzh.ch/site/docs/owl\_to\_ace.html`. A domain-specific version of SWAT for the SNOMED biomedical ontology has also been developed (Liang et al., 2011a; Liang et al., 2011b).

[26]The verbalizer also organizes the English-like sentences of each glossary entry under sub-headings like 'Definition', 'Taxonomy', 'Description', 'Distinctions' (Williams et al., 2011). We discarded these subheadings, whose meanings were not entirely clear to us, but we retained the order of the sentences.

because the development of the domain-dependent generation resources was combined with the development of the ontologies. Since the time needed to create the domain-dependent generation resources depends on one's familiarity with NaturalOWL and its Protégé plug-in, exact times are not particularly informative. Instead, we report figures such as the number of sentence plans, lexicon entries etc. that were required, along with approximate times. We do not evaluate the usability of the Protégé plug-in of NaturalOWL, since it is very similar to the authoring tool of M-PIRO. Previous experiments (Androutsopoulos et al., 2007) showed that computer science graduates with no expertise in NLG could learn to use effectively the authoring tool of M-PIRO to create the necessary domain-dependent generation resources for existing or new ontologies, after receiving the equivalent of a full-day introduction course.

2.4.1 Trials with the Wine Ontology

In the first trial, we experimented with the Wine Ontology, which is often used in Semantic Web tutorials.[27] It comprises 63 wine classes, 52 wine individuals, a total of 238 classes and individuals (including wineries, regions, etc.), and 14 properties. The Wine Ontology contains axioms and expressions covering almost all the functionality of OWL 2, making it a very appropriate test case for NaturalOWL's capabilities.

We submitted the Wine Ontology to the SWAT verbalizer to obtain its glossary of English-like descriptions of classes, properties, and individuals. We retained only the descriptions of the 63 wine classes and the 52 wine individuals. Subsequently, we also discarded 20 of the 63 wine class descriptions, as they were for trivial classes (e.g., `RedWine`) and they were stating the obvious (e.g., "A red wine is defined as a wine that has as color Red").[28] In the descriptions of the remaining 43 wine classes and

[27] See `http://www.w3.org/TR/owl-guide/wine.rdf`.

[28] Third (2012) discusses how OWL axioms leading to undesirable sentences of this kind might be detected.

52 wine individuals, we discarded sentences expressing axioms that NaturalOWL does not consider, for example sentences providing examples of individuals that belong in a class being described. The remaining sentences express the same OWL statements that NaturalOWL expresses when its maximum fact distance is set to 1. Two examples of texts produced by the SWAT verbalizer follow.

> Chenin Blanc (class): A chenin blanc is defined as something that is a wine, is made from grape the Chenin Blanc Grape, and is made from grape at most one thing. A chenin blanc both has as flavor Moderate, and has as color White. A chenin blanc both has as sugar only Off Dry and Dry, and has as body only Full and Medium.

> The Foxen Chenin Blanc (individual): The Foxen Chenin Blanc is a chenin blanc. The Foxen Chenin Blanc has as body Full. The Foxen Chenin Blanc has as flavor Moderate. The Foxen Chenin Blanc has as maker Foxen. The Foxen Chenin Blanc has as sugar Dry. The Foxen Chenin Blanc is located in the Santa Barbara Region.

Afterwards, we generated texts for the 43 classes and 52 individuals using NaturalOWL without domain-dependent generation resources, hereafter called NaturalOWL$(-)$, setting the maximum fact distance to 1; the resulting texts were very similar to SWAT's.

We then constructed the domain-dependent generation resources of NaturalOWL for the Wine Ontology. The resources are summarized in Table 2.4. They were constructed by the author of this thesis, who devoted three days to their construction, testing, and refinement.[29] Our experience is that it takes weeks (if not longer) to develop an OWL ontology the size of the Wine Ontology (acquire domain knowledge, formulate the axioms in OWL, check for inconsistencies, populate the ontology with individuals etc.); hence, a period of a few days is relatively light effort, compared to the time needed to develop

[29] Some of the resources were constructed by editing directly their OWL representations, rather than using the Protégé plug-in, which was not fully functional at that time. By using the now fully functional plug-in, the time to create the domain-dependent generation resources would have been shorter.

Resources	English	Greek
Sections	2	
Property assignments to sections	7	
Interest score assignments	8	
Sentence plans	5	–
Lexicon entries	67	–
Natural language names	41	–

Table 2.4: Domain-dependent generation resources created for the Wine Ontology.

an OWL ontology of this size. Only English texts were generated in this trial; hence, no Greek resources were constructed. We defined only one user type, and we used interest scores only to block sentences stating the obvious, by assigning zero interest scores to the corresponding message triples; we also set `maxMessagesPerSentence` to 3.

Only 7 of the 14 properties of the Wine Ontology are used in the OWL statements that describe the 43 classes and 52 individuals. We defined only 5 sentence plans, as some of the 7 properties could be expressed by the same sentence plans. We did not define multiple sentence plans per property. We also assigned the 7 properties to 2 sections, and ordered the sections and properties. We created NL names only when the automatically extracted ones were causing disfluencies. The extracted NL names were obtained from the OWL identifiers of classes and individuals; no `rdfs:label` strings were available. To reduce the number of manually constructed NL names further, we declared the 52 individual wines to be anonymous (and provided no NL names for them). Most of the 67 lexicon entries were used in the remaining 41 NL names of classes and individuals; NL names were very simple, having 2 slots on average. We used NaturalOWL with the domain-dependent resources, hereafter called NaturalOWL(+), to re-generate the 95 texts, again setting the maximum fact distance to 1; example texts follow.

Chenin Blanc (class): A Chenin Blanc is a moderate, white wine. It has only a full or medium body. It is only off-dry or dry. It is made from exactly one wine grape variety: Chenin Blanc grapes.

The Foxen Chenin Blanc (individual): This wine is a moderate, dry Chenin Blanc. It has a full body. It is made by Foxen in the Santa Barbara County.

The resulting 285 texts (95×3) of the three systems we examine (SWAT verbalizer, NaturalOWL$(-)$, NaturalOWL$(+)$) were shown to 10 computer science students (both undergraduates and graduate students), who were not involved in the development of NaturalOWL; they were all fluent in English, though not native English speakers, and they did not consider themselves wine experts. The students were told that a glossary of wines was being developed for people who were interested in wines and knew basic wine terms (e.g., wine colors, wine flavors), but who were otherwise not wine experts. Each one of the 285 texts was given to exactly one student. Each student was given approximately 30 texts, approximately 10 randomly selected texts from each system. The OWL statements that the texts were generated from were not shown, and the students did not know which system had generated each text. Each student was shown all of his/her texts in random order, regardless of the system that had generated them. The students were asked to score each text by stating how strongly they agreed or disagreed with statements S_1–S_5 below. A scale from 1 to 3 was used (1: disagreement, 2: ambivalent, 3: agreement).

(S_1) *Sentence fluency*: The sentences of the text are fluent, i.e., each sentence *on its own* is grammatical and sounds natural. When two or more smaller sentences are combined to form a single, longer sentence, the resulting longer sentence is also grammatical and sounds natural.

(S_2) *Referring expressions*: The use of pronouns and other referring expressions (e.g., "this wine") is appropriate. The choices of referring expressions (e.g., when to use a pronoun or other expression instead of the name of an object) sound natural, and it is easy to understand what these expressions refer to.

(S_3) *Text structure*: The order of the sentences is appropriate. The text presents information by moving reasonably from one topic to another.

Criteria	SWAT	NaturalOWL(−)	NaturalOWL(+)
Sentence fluency	2.00 ± 0.15	1.76 ± 0.15	$\mathbf{2.80 \pm 0.10}$
Referring expressions	1.40 ± 0.13	1.15 ± 0.09	$\mathbf{2.72 \pm 0.13}$
Text structure	2.15 ± 0.16	2.20 ± 0.16	$\mathbf{2.94 \pm 0.05}$
Clarity	2.66 ± 0.13	2.55 ± 0.13	$\mathbf{2.74} \pm 0.11$
Interest	2.30 ± 0.15	2.14 ± 0.16	$\mathbf{2.68 \pm 0.12}$

Table 2.5: Results for texts generated from the Wine Ontology by the SWAT verbalizer and NaturalOWL with (+) and without (−) domain-dependent generation resources.

(S_4) *Clarity*: The text is easy to understand, provided that the reader is familiar with basic wine terms.

(S_5) *Interest*: People interested in wines, but who are not wine experts, would find the information interesting. Furthermore, there are no redundant sentences in the text (e.g., sentences stating the obvious).[30]

S_5 assesses *content selection*, the first processing sub-stage; we expected the differences across the three systems to be very small, as they all reported the same information, with the exception of redundant sentences blocked by using zero interest assignments in NaturalOWL. S_3 assesses *text planning*, the second sub-stage; again we expected small differences, as many of the wine properties can be mentioned in any order, though there are some properties (e.g., maker, location) that are most naturally reported separately from others (e.g., color, flavor), which is why we used two sections (Table 2.4). S_1 assesses *lexicalization* and *aggregation*; we decided not to use separate statements for these two stages, since it might have been difficult for the students to understand exactly when aggregation takes place. S_2 assesses *referring expression* generation. S_4 measures the overall perceived clarity of the texts. There was no statement for *surface realization*, as this stage had a rather trivial effect.

[30]The students were told not to consider whether or not *additional* information should have been included.

Table 2.5 shows the average scores of the three systems, with averages computed on the 95 texts of each system, along with 95% confidence intervals (of sample means). For each criterion, the best score is shown in bold; the confidence interval of the best score is also shown in bold if it does not overlap with the other confidence intervals.[31]

As expected, the domain-dependent generation resources clearly help NaturalOWL produce more fluent sentences and much better referring expressions. The text structure scores show that the assignment of the ontology's properties to sections and the ordering of the sections and properties had a greater impact on the perceived structure of the texts than we expected. The highest score of the SWAT verbalizer was obtained in the clarity criterion, which agrees with our experience that one can usually understand what the texts of the SWAT verbalizer mean, even if their sentences are often not entirely fluent, not particularly well ordered, and keep repeating proper names. NaturalOWL$(+)$ had the highest clarity score, but the difference from the SWAT verbalizer, which had the second highest score, is not statistically significant. NaturalOWL$(+)$ also obtained higher interest scores than the other two systems, with statistically significant differences from both; these differences, which are larger than we expected, can only be attributed to the zero interest score assignments of the domain-dependent generation resources, which blocked sentences stating the obvious, because otherwise all three systems report the same information.

The SWAT verbalizer obtained higher scores than NaturalOWL$(-)$, with the text structure score being the only exception. Only the difference in the referring expression scores of the two systems, though, was found to be statistically significant. Both systems, however, received particularly low scores for their referring expressions, which is

[31] When two intervals do not overlap, the difference is statistically significant. When they overlap, the difference may still be statistically significant; we performed paired two-tailed t-tests ($\alpha = 0.05$) in these cases. In a pilot study, we also measured the inter-annotator agreement of two of the students on a sample of 30 texts (10 from each system). Agreement was very high (sample Pearson correlation $r \geq 0.91$) in all five criteria. A similar pilot study was performed in the next trial, also indicating very high agreement.

not surprising, given that they both always refer to individuals and classes by extracted names; the slightly higher score of the SWAT verbalizer is probably due to its better tokenization of OWL identifiers.

2.4.2 Trials with the Consumer Electronics Ontology

In the second trial, we experimented with the Consumer Electronics Ontology, an OWL ontology for consumer electronics products and services.[32] The ontology comprises 54 classes and 441 individuals (e.g., printer types, paper sizes, manufacturers), but no information about particular products. We added 60 individuals describing 20 digital cameras, 20 camcorders, and 20 printers. The 60 individuals were randomly selected from a publicly available dataset of 286 digital cameras, 613 camcorders, and 58 printers, whose instances comply with the Consumer Electronics Ontology.[33]

We submitted the Consumer Electronics Ontology with the additional 60 individuals to the SWAT verbalizer, and retained only the descriptions of the 60 individuals. Again, we removed sentences expressing axioms NaturalOWL does not consider. We also renamed the string values of some datatype properties to make the texts easier to understand (e.g., "CMT" became "cm"). An example description follows.

> The Sony Cyber-shot DSC-T90 is a digital camera.
>
> The Sony Cyber-shot DSC-T90 has as manufacturer Sony.
>
> The Sony Cyber-shot DSC-T90 has as data interface type Usb2 0.
>
> The Sony Cyber-shot DSC-T90 has as depth Depth. Depth has as unit of measurement cm. Depth has as value float 9.4.
>
> The Sony Cyber-shot DSC-T90 has as digital zoom factor the Digital Zoom Factor. The Digital Zoom Factor has as value float 12.1. [...]
>
> The Sony Cyber-shot DSC-T90 has as feature Video Recording, Microphone and the Automatic Picture Stabilizer.

[32] `http://www.ebusiness-unibw.org/ontologies/consumerelectronics/v1`.

[33] See `http://rdf4ecommerce.esolda.com/` for the dataset that we used. A list of similar datasets is available at `http://wiki.goodrelations-vocabulary.org/Datasets`.

The Sony Cyber-shot DSC-T90 has as self timer true. [...]

In this ontology, many properties have composite values, expressed by using auxiliary individuals. In the example above, a property (`hasDepth`) connects the digital camera to an auxiliary individual `Depth` (similar to the anonymous node `_:n` of the property concatenation price example of page 41), which is then connected via two other properties (`hasValueFloat` and `hasUnitOfMeasurement`) to the float value 9.4 and the unit of measurement (centimeters), respectively. We obtained the descriptions of the auxiliary individuals (e.g., `Depth`), which are different entries in the glossary of the SWAT verbalizer, and we copied them immediately after the corresponding sentences that introduce the auxiliary individuals. We also formatted each text as a list of sentences, as above, to improve readability.

We then generated texts for the 60 products using NaturalOWL$(-)$, setting the maximum fact distance to 1. Descriptions of auxiliary individuals were also generated and copied immediately after the sentences introducing them. The texts were very similar to those of the SWAT verbalizer, and they were formatted in the same manner.

In this trial, we also wanted to consider a scenario where the set of individuals to be described changes frequently (e.g., the products sold by a reseller change, new products arrive etc.) along with changes in other connected individuals (e.g., new manufacturers may be added), but nothing else in the ontology changes, i.e., only the assertional knowledge changes. In this case, it may be impractical to update the domain-dependent generation resources whenever the population of individuals changes. Our hypothesis was that by considering a sample of individuals of the types to be described (printers, cameras, camcorders, in our case), it would be possible to construct domain-dependent generation resources (e.g., sections, the ordering of sections and properties, sentence plans, the NL names of classes) that would help NaturalOWL generate reasonably good descriptions of new (unseen) individuals (products), without updating the domain-dependent generation resources, using the tokenized OWL identifiers or

`rdfs:label` strings of the new individuals as their NL names.

To simulate this scenario, we randomly split the 60 products in two non-overlapping sets, the *development set* and the *test set*, each consisting of 10 digital cameras, 10 camcorders, and 10 printers. Again, the author of this thesis constructed and refined the domain-dependent generation resources of NaturalOWL, this time by considering a version of the ontology that included the 30 development products, but not the 30 test products, and by viewing the generated texts of the 30 development products only. This took approximately six days (for two languages).[34] Hence, relatively light effort was again needed, compared to the time it typically takes to develop an ontology of this size, with terminology in two languages. Texts for the 30 products of the test set were then also generated by using NaturalOWL and the domain-dependent generation resources of the development set.

As in the previous trial, we defined only one user type, and we used interest scores only to block sentences stating the obvious. The maximum messages per sentence was again 3. We constructed domain-dependent generation resources for both English and Greek; the resources are summarized in Table 2.6. We created sentence plans only for the 42 properties of the ontology that were used in the development set (one sentence plan per property); the test set uses two additional properties, for which the default sentence plans of NaturalOWL (for English and Greek) were used. We also assigned the 42 properties to 6 sections, and ordered the sections and properties. We created NL names only when the automatically extracted ones were causing disfluencies in the development texts. Unlike the previous trial, the products to be described were not declared to be anonymous individuals, but the number of NL names that had to be provided was roughly the same as in the previous trial, since fewer automatically

[34] Again, some of the domain-dependent generation resources were constructed by editing their OWL representations. As a test, the author of this thesis later reconstructed the domain-dependent generation resources from scratch using the fully functional Protégé plug-in, this time in four days.

Resources	English	Greek
Sections	6	
Property assignments to sections	42	
Interest score assignments	12	
Sentence plans	42	42
Lexicon entries	19	19
Natural language names	36	36

Table 2.6: Domain-dependent generation resources for the Consmer Electronics Ontology

extracted names were causing disfluencies; in particular, all the products had reasonably good `rdfs:label` strings providing their English names.

The following is an example description from the development set generated by NaturalOWL(+). We formatted the sentences of each section as a separate paragraph, headed by the name of the section (e.g., "Other features:"); this was easy, because NaturalOWL can automatically mark up the sections in the texts. The maximum fact distance was again 1, but the sentence plans caused NaturalOWL to automatically retrieve additional message triples describing the auxiliary individuals at distance 1; hence, we did not have to retrieve this information manually, unlike the texts of the SWAT verbalizer and NaturalOWL(−).

Type: Sony Cyber-shot DSC-T90 is a digital camera.

Main features: It has a focal length range of 35.0 to 140.0 mm, a shutter lag of 2.0 to 0.0010 sec and an optical zoom factor of 4.0. It has a digital zoom factor of 12.1 and its display has a diagonal of 3.0 in.

Other features: It features an automatic picture stabilizer, a microphone, video recording and it has a self-timer.

Energy and environment: It uses batteries.

Connectivity, compatibility, memory: It supports USB 2.0 connections for data exchange and it has an internal memory of 11.0 GB.

Dimensions and weight: It is 5.7 cm high, 1.5 cm wide and 9.4 cm deep. It weighs 128.0 grm.

The 180 English texts that were generated by the three systems for the 30 development and 30 test products were shown to the same 10 students of the first trial. The students were now told that the texts would be used in on-line descriptions of products in the Web site of a retailer. Again, the OWL statements that the texts were generated from were not shown to the students, and the students did not know which system had generated each text. Each student was shown 18 randomly selected texts, 9 for products of the development set (3 texts per system) and 9 for products of the test set (again 3 texts per system). Each student was shown all of his/her texts in random order, regardless of the system that had generated them. The students were asked to score the texts as in the previous trial.

Table 2.7 shows the results for the English texts of the *development* set.[35] As in the previous trial, the domain-dependent generation resources clearly help NaturalOWL produce much more fluent sentences, and much better referring expressions and sentence orderings. The text structure scores of the SWAT verbalizer and NaturalOWL$(-)$ are now much lower than in the previous trial, because there are now more message triples to express per individual and more topics, and the texts of these systems jump from one topic to another making the texts look very incoherent; for example, a sentence about the width of a camera may be separated from a sentence about its height by a sentence about shutter lag. This incoherence may have also contributed to the much lower clarity scores of these two systems, compared to the previous trial. The interest scores of these two systems are also much lower than in the previous trial; this may be due to the verbosity of their texts, caused by their frequent references to auxiliary individuals in the second trial, combined with the lack (or very little use) of sentence aggregation

[35] When a confidence interval is 0.00, this means that all the students gave the same score to all texts.

Criteria	SWAT	NaturalOWL(−)	NaturalOWL(+)
Sentence fluency	1.97 ±0.15	1.93 ±0.27	**2.90 ±0.08**
Referring expressions	1.10 ±0.06	1.10 ±0.11	**2.87 ±0.08**
Text structure	1.67 ±0.15	1.33 ±0.19	**2.97 ±0.04**
Clarity	1.97 ±0.15	2.07 ±0.26	**3.00 ±0.00**
Interest	1.77 ±0.14	1.73 ±0.29	**3.00 ±0.00**

Table 2.7: English development results for the Consumer Electronics Ontology.

and pronoun generation. By contrast, the clarity and interest of NaturalOWL(+) were judged to be perfect; the poor clarity and interest of the other two systems may have contributed to these perfect scores though. Again, the SWAT verbalizer obtained slightly better scores than NaturalOWL *without* domain-dependent generation resources, except for clarity, but the differences are not statistically significant.

Table 2.8 shows the results for the English texts of the *test* set. The results of the SWAT verbalizer and NaturalOWL(−) are very similar to those of Table 2.7, as one would expect. Also, there was only a very marginal decrease in the scores of NaturalOWL(+), compared to the scores of the same system for the development set in Table 2.7. No statistically significant difference was detected, however, between the corresponding cells of the two tables, for any of the three systems. These results support our hypothesis that by considering a sample of individuals of the types to be described one can construct domain-dependent generation resources that can be used to produce high-quality texts for new individuals of the same types, when the rest of the ontology remains unchanged. The fact that all the products (but not the other individuals) had `rdfs:label` strings providing their English names probably contributed to the results of NaturalOWL(+) in the test set, but `rdfs:label` strings of this kind are common in OWL ontologies.

We then showed the 60 Greek texts that were generated by NaturalOWL(+) to the same 10 students, who were native Greek speakers; the SWAT verbalizer and Natu-

Criteria	SWAT	NaturalOWL(−)	NaturalOWL(+)
Sentence fluency	2.03 ±0.15	1.87 ±0.15	**2.87 ±0.08**
Referring expressions	1.10 ±0.06	1.10 ±0.06	**2.87 ±0.08**
Text structure	1.57 ±0.13	1.37 ±0.12	**2.93 ±0.05**
Clarity	2.07 ±0.15	1.93 ±0.15	**2.97 ±0.04**
Interest	1.83 ±0.17	1.60 ±0.14	**2.97 ±0.04**

Table 2.8: English test results for the Consumer Electronics Ontology.

Criteria	NaturalOWL(+), development data	NaturalOWL(+), test data
Sentence fluency	2.87 ±0.12	2.83 ±0.09
Referring expressions	2.77 ±0.20	2.80 ±0.11
Text structure	3.00 ±0.00	3.00 ±0.00
Clarity	3.00 ±0.00	2.93 ±0.05
Interest	2.97 ±0.06	3.00 ±0.00

Table 2.9: Greek results for the Consumer Electronics Ontology.

ralOWL(−) cannot generate Greek texts from the Consumer Electronics ontology. Table 2.9 shows the results we obtained for the Greek texts of the development and test sets. There is no statistically significant difference from the corresponding results for English (cf. the last columns of Tables 2.7 and 2.8). No statistically significant difference was detected in the results for the Greek texts of the development and test sets (Table 2.9). We note, however, that it is common to use English names of electronics products in Greek texts, which made using the English `rdfs:label` names of the products in the Greek texts acceptable. In other domains, for example cultural heritage, it might be unacceptable to use English names of individuals; hence, one would have to provide Greek NL names for new individuals.

2.4.3 Trials with the Disease Ontology

In the third trial, we experimented with the Disease Ontology, an OWL ontology which describes diseases, including their symptoms, causes etc.[36] The ontology contains information about 6,286 diseases, all represented as classes. Apart from IS-A and synonym properties, however, all the other information in the ontology is represented using strings containing quasi-English sentences with property names used mostly as verbs. For example, there is an axiom in the ontology stating that the Rift Valley Fever (DOID_1328) is a kind of viral infectious disease (DOID_934). There are also cross-references to equivalent or related concepts of other biomedical ontologies (e.g., UMLS) and occasionally pointers to subsets (broader classes) of diseases.[37] All the other information, in our example about the Rift Valley Fever, is provided in a string, shown below as 'Definition'. The tokens that contain underscores (e.g., `results_in`) are property names. The ontology declares all the property names, but uses them only

[36] See `http://disease-ontology.org/`. This section reports work jointly carried out with M. Evaggelakaki for her BSc thesis (Evaggelakaki, 2014).

[37] Consult `http://www.nlm.nih.gov/research/umls/` for information on UMLS.

inside 'Definition' strings. Apart from diseases, it does not define any of the other entities mentioned in the 'Definition' strings (e.g., symptoms, viruses).

Name: Rift Valley Fever (DOID_1328)

IS-A: viral infectious disease (DOID_934)

Definition: A viral infectious disease that `results_in` infection, `has_material_basis_in` Rift Valley fever virus, which is `transmitted_by` Aedes mosquitoes. The virus affects domestic animals (cattle, buffalo, sheep, goats, and camels) and humans. The infection `has_symptom` jaundice, `has_symptom` vomiting blood, `has_symptom` passing blood in the feces, `has_symptom` ecchymoses (caused by bleeding in the skin), `has_symptom` bleeding from the nose or gums, `has_symptom` menorrhagia and `has_symptom` bleeding from venepuncture sites.

We modified the Disease Ontology, defining as individuals all the non-disease entities mentioned in the 'Definition' strings, and adding statements to formally express the properties mentioned in the original 'Definition' strings. For example, the resulting OWL ontology contains the following definition of Rift Valley Fever, where `infection`, `Rift_Valley_fever_virus`, `Aedes_mosquitoes`, `jaundice`, etc. are new individuals.

```
SubClassOf(:DOID_1328
   ObjectIntersectionOf(:DOID_934
      ObjectHasValue(:results_in :infection)
      ObjectHasValue(:has_material_basis_in :Rift_Valley_fever_virus)
      ObjectHasValue(:transmitted_by :Aedes_mosquitoes)
      ObjectHasValue(:has_symptom :jaundice)
      ObjectHasValue(:has_symptom :vomiting_blood)
      ObjectHasValue(:has_symptom :passing_blood_in_the_feces)
      ObjectHasValue(:has_symptom :ecchymoses_(caused_by_bleeding_in_the_skin))
      ObjectHasValue(:has_symptom :bleeding_from_the_nose_or_gums)
      ObjectHasValue(:has_symptom :menorrhagia)
      ObjectHasValue(:has_symptom :bleeding_from_venepuncture_sites)))
```

The new form of the ontology was produced automatically, using patterns that searched the definition strings for declared (in the original form of the ontology) property names

(e.g., `results_in`), sentence breaks, and words that introduce secondary clauses (e.g., "that", "which").[38] Some sentences of the original definition strings that did not include declared property names, like the sentence "The virus affects. . . humans" in the 'Definition' string of Rift Valley Fever above, were discarded during the conversion, because it was not always possible to reliably convert them to appropriate OWL statements.

The new form of the Disease Ontology contains 6,746 classes, 15 properties, and 1,545 individuals. We ignore the 5,014 classes which participate only in IS-A and synonym properties, since their generated descriptions would not be particularly interesting. From the remaining 1,732, we randomly selected 400 disease classes. We did not submit the classes of the Disease Ontology to the SWAT verbalizer, but instead used the 'Definition' strings to compare against; their text quality is roughly equivalent to that of a verbalizer and we also wished to measure how much information in the 'Definition' strings is missed by our automatic conversion.

We then generated texts for the 200 diseases using NaturalOWL$(-)$, setting the maximum fact distance to 1. To simulate a scenario similar to the one described in the previous section, we randomly split the 400 classes in two non-overlapping sets, the *development set* and the *test set*, each consisting of 200 diseases. This time the domain-dependent generation resources of NaturalOWL were constructed and refined by a computer science undergraduate student, who was not involved in the development of NaturalOWL; the student was fluent in English, was given the equivalent of a full-day introduction course to NaturalOWL and the authoring plug-in, and constructed the generation resources considering a version of the ontology that included the 200 development diseases (and their generated texts), but not the 200 test diseases. This took approximately 5 days.[39] Hence, relatively light effort was again needed, especially considering the relative non-expertise of the student. Texts for the 200 classes of the test set

[38]The OWL version of the Disease Ontology that we produced is available upon request.

[39]These domain-dependent generation resources were constructed using the fully functional Protégé plug-in.

Resources	English	Greek
Sections	4	
Property assignments to sections	8	
Interest score assignments	102	
Sentence plans	8	–
Lexicon entries	386	–
Natural language names	397	–

Table 2.10: Domain-dependent generation resources created for the Disease Ontology.

were then also generated by using NaturalOWL and the domain-dependent generation resources of the development set.

Similar to the previous trial, we defined only one user type, and we used interest scores only to block sentences stating the obvious; the maximum messages per sentence was again set to 3. The resources are summarized in Table 2.10. We created sentence plans only for the 8 properties of the ontology that were used in the development set (one sentence plan per property); the test set uses no additional properties. We also assigned the 8 properties to 4 sections, and ordered the sections and properties according to the guidelines of the Disease Ontology.[40] As in the Consumer Electronics Ontology, we created NL names only when the automatically extracted ones were causing disfluencies in the development texts; no diseases were declared to be anonymous individuals.

An example description from the development set by NaturalOWL(+) follows.

> Rift valley fever is a kind of viral infectious disease that results in infections. Its symptoms are vomiting blood, bleeding from the nose, jaundice, menorrhagia, passing blood in the feces, ecchymoses (caused by bleeding in the skin) and bleeding from venepuncture sites. It is transmitted by the aedes mosquitoes.

[40] The guidelines are available at `http://do-wiki.nubic.northwestern.edu/do-wiki/index.php/Style_Guide`.

The 1140 texts that were generated by the three systems for 190 randomly selected development diseases and 190 randomly selected test diseases were shown to two biomedical experts, both fluent in English.[41] The experts were told that the texts would be used in on-line descriptions of diseases in a biomedical Web directory. The OWL statements that the texts were generated from were not shown to the experts, and the experts did not know which system had generated each text. Each expert was shown 570 randomly selected texts, 285 for diseases of the development set (95 texts per system) and 285 for diseases of the test set (again 95 texts per system). For each disease, the three texts were shown side by side in random order, and the experts were instructed to take into account the differences of the three texts. The experts were asked to score each text by stating how strongly they agreed or disagreed with statements S_1–S_3 below. A scale from 1 to 5 was used (1: strong disagreement, 2: disagreement, 3: ambivalent, 4: agreement, 5: strong agreement).

(S_1) *Readability*: The sentences of the text are fluent, i.e., each sentence *on its own* is grammatical and sounds natural. Each sentence is also easy to understand, provided that the reader is familiar with basic biomedical terms.

(S_2) *Structure*: The order of the sentences is appropriate and according to the guidelines of the Disease ontology.

(S_3) *Informativeness*: The most informative text of the three should get the maximum score, with the other two texts getting lower scores depending on how much less information they express.

S_1 assesses *lexicalization*, *aggregation* and the overall perceived clarity of the texts. S_2 assesses *text planning*; we wished to measure whether the 'Definition' strings which are human-authored follow the ontology's guidelines as well as our systems do. S_3 assesses the amount of information lost by our automatic conversion of the ontology.

[41] The remaining 60 texts (20 from each system) for 10 development and 10 test diseases were used to measure the inter-annotator agreement of the two experts. Agreement was high (sample Pearson correlation $r \geq 0.38$, sample Spearman correlation $\rho \geq 0.87$, sample Kendall correlation $\tau \geq 0.82$) in all three criteria.

Criteria	DEFINITION	NaturalOWL$(-)$	NaturalOWL$(+)$
Readability	2.78 ± 0.21	3.85 ± 0.21	$\mathbf{4.96 \pm 0.13}$
Structure	3.64 ± 0.22	4.22 ± 0.23	$\mathbf{4.73 \pm 0.10}$
Informativeness	$\mathbf{4.91 \pm 0.15}$	4.73 ± 0.06	4.72 ± 0.13

Table 2.11: Development results for the Disease Ontology.

Criteria	DEFINITION	NaturalOWL$(-)$	NaturalOWL$(+)$
Readability	2.55 ± 0.24	3.66 ± 0.25	$\mathbf{4.89 \pm 0.16}$
Structure	3.79 ± 0.22	4.11 ± 0.26	$\mathbf{4.78 \pm 0.06}$
Informativeness	$\mathbf{4.98 \pm 0.15}$	4.65 ± 0.12	4.65 ± 0.16

Table 2.12: Test results for the Disease Ontology.

Table 2.11 shows the results for the texts of the *development* set. As in the previous trials, the domain-dependent generation resources clearly help NaturalOWL produce much more readable sentences. The text structure scores of the 'Definition' strings are lower than the scores of our two systems, since the automatic systems can follow the guidelines more closely. The informativess scores show that our automatic conversion of the Disease Ontology misses a rather small amount of information.

Table 2.12 shows the results for the texts of the *test* set which are very similar to those of Table 2.11. These results further support our hypothesis of the previous section, that by considering a sample of classes one can construct domain-dependent generation resources that can be used to produce high-quality texts for new classes. Similarly to the Consumer Electronics Ontology's products, the fact that all the diseases had `rdfs:label` strings providing their names (almost entirely grammatically correct with occasionally missing articles or prepositions) probably contributed to the high results of NaturalOWL$(+)$ in the test set.

2.4.4 Ablation trials with the Consumer Electronics Ontology

In the last trial, we studied how the quality of the generated texts is affected when various components and domain-dependent generation resources of NaturalOWL are gradually removed. We used the Consumer Electronics Ontology, with the domain-dependent generation resources that we had constructed for the 30 development products of the previous trial. We also used 45 new test products (15 digital cameras, 15 camcorders, and 15 printers, from the same publicly available dataset), other than the 30 development and the 30 test products of the previous trial.

We generated English texts for the 45 new test products, using the 7 configurations of NaturalOWL of Table 2.13. The resulting $45 \times 7 = 315$ texts were shown to 7 students, who had the same background as in the previous trials. Each student was shown the 7 texts of 6 or 7 test products (42 or 49 texts per student). For each product, the 7 texts were shown side by side in random order, and the students were instructed to take into account the differences of the 7 texts. The students did not know which system had generated which text. Statements S_1–S_5 of Section 2.4.1 were used again as criteria, but a scale from 1 to 5 was used this time (1: strong disagreement, 2: disagreement, 3: ambivalent, 4: agreement, 5: strong agreement), to make it easier to distinguish between the 7 configurations.

The first configuration (NaturalOWL$(+)$) is NaturalOWL with all of its components enabled, using all the available domain-dependent generation resources. As in the previous corresponding trial (see Table 2.8), the texts of this configuration were judged to be near-perfect by all the criteria. The second configuration was the same, but without the interest score assignments. The results of the second configuration were very close to the results of the first one, since interest score assignments were used only to avoid generating sentences stating the obvious (e.g., "Sony Cyber-shot DSC-T90 is manufactured by Sony"). The biggest decrease was in the interest criterion, as one would expect, but the scores for sentence fluency and clarity were also affected, presumably because

No.	System Configuration	Sentence Fluency	Ref. Expressions	Text Structure	Clarity	Interest
1	NaturalOWL(+)	**4.80 ±0.12**	**5.00 ±0.00**	**4.82 ±0.15**	**4.78 ±0.12**	**4.89 ±0.09**
2	− interest scores	4.53 ±0.16	4.95 ±0.06	4.78 ±0.12	4.62 ±0.17	4.20 ±0.19
3	− ref. expr. gen.	3.93 ±0.28	*1.53 ±0.22*	4.80 ±0.12	4.51 ±0.24	4.07 ±0.22
4	− NL names	3.71 ±0.29	1.48 ±0.21	4.71 ±0.15	4.24 ±0.25	3.98 ±0.26
5	− aggregation	3.64 ±0.33	1.33 ±0.19	4.67 ±0.16	4.24 ±0.25	3.93 ±0.26
6	− sentence plans	*2.07 ±0.37*	1.33 ±0.19	4.60 ±0.18	*2.49 ±0.36*	*2.38* ±0.35
7	− sections, ordering	1.89 ±0.36	1.33 ±0.19	*1.53 ±0.24*	2.33 ±0.33	1.89 ±0.28

Table 2.13: Ablation English test results for the Consumer Electronics Ontology. Each configuration removes a component or resource from the previous configuration.

the sentences that state the obvious sound unnatural and seem to introduce noise. There were very small differences in the scores for referring expressions and text structure, which seem to suggest that when the overall quality of the texts decreases, the judges are biased towards assigning lower scores in all of the criteria.[42]

The third configuration was the same as the second one, but the component that generates pronouns and demonstrative noun phrases was disabled, causing NaturalOWL to always use the NL names of the individuals and classes, or names extracted from the ontology. There was a big decrease in the score for referring expressions, showing that despite their simplicity, the referring expression generation methods of NaturalOWL have a noticeable effect; we mark big decreases in italics in Table 2.13. The scores for sentence fluency, interest, and clarity were also affected, presumably because repeating the names of the individuals and classes made the sentences look less natural, boring, and more difficult to follow. There was almost no difference (a very small positive one) in the text structure score.

[42] In all of the criteria, all the differences from one configuration to the next one are statistically significant, with the only exceptions being the differences in clarity between configurations 4 and 5, and the differences in the scores for referring expressions between configurations 5–6 and 6–7. Again, when the 95% confidence intervals overlapped, we performed paired two-tailed t-tests ($\alpha = 0.05$).

In the fourth configuration, the NL names of the individuals and classes were also removed, forcing NaturalOWL to always use automatically extracted names. There was a further decrease in the score for referring expressions, but the decrease was small, because the referring expressions were already poor in the third configuration. Note, also, that the NL names are necessary for NaturalOWL to produce pronouns and demonstrative noun phrases; hence, the higher referring expression score of the third configuration would not have been possible without the NL names. The sentence fluency and clarity scores were also affected in the fourth configuration, presumably because the automatically extracted names made the texts more difficult to read and understand. There were also small decreases in the scores for interest and even text structure, suggesting again that when the overall quality of the texts decreases, the judges are biased towards lower scores in all of the criteria.

In the fifth configuration, aggregation was turned off, causing NaturalOWL to produce a separate sentence for each message triple. With sentences sharing the same subject no longer being aggregated, more referring expressions for subjects had to be generated. Since the component that generates pronouns and demonstrative noun phrases had been switched off and the NL names had been removed, more repetitions of automatically extracted names had to be used, which is why the score for referring expressions decreased further. Sentence fluency was also affected, since some obvious aggregations were no longer being made, which made the sentences look less natural. There was also a small decrease in the score for the perceived text structure and interest, but no difference in the score for clarity. Overall, the contribution of aggregation to the perceived quality of the texts seems to be rather small.

In the sixth configuration, all the sentence plans were removed, forcing NaturalOWL to use the default sentence plan and tokenized property identifiers. There was a sharp decrease in sentence fluency and clarity, as one would expect, but also in the perceived interest of the texts. There was also a small decrease in the perceived text structure, and

no difference in the score for referring expressions. Overall, these results indicate that sentence plans are a very important part of the domain-dependent generation resources.

In the seventh configuration, the sections, assignments of properties to sections, and the ordering of sections and properties were removed, causing NaturalOWL to produce random orderings of the message triples. There was a very sharp decrease in the score for text structure. The scores for the perceived interest, clarity, but also sentence fluency were also affected, again suggesting that when the overall quality of the texts decreases, the judges are biased towards lower scores in all of the criteria.

We conclude that the sections and ordering information of the domain-dependent generation resources are, along with the sentence plans, particularly important. The NL names can also be considered particularly important, since they affect the generation of referring expressions as well. We note, however, that the best scores were obtained by enabling all the components and using all the available domain-dependent generation resources.

2.5 Differences between NaturalOWL version 1 and 2

This chapter described version 2 of NaturalOWL, which is the version developed during the work of this thesis and extends version 1 (Galanis and Androutsopoulos, 2007). This section summarizes the main differences between the two versions.

Version 2 supports OWL 2 ontologies, while version 1 supports only the previous version of OWL. Version 2 also supports ontologies that import resources from other ontologies; the Wine and Consumer Electronics ontologies are both examples of this. The new version can also produce texts describing classes instead of only individuals (Section 2.3.1.1), even when the class definitions involve complex axioms (e.g., <*Target*, minCardinality(*prop*), *number*[:*NamedClass*]>; see Tables 2.1 and 2.2).

We have also extended the domain-dependent linguistic resources that version 1 of

NaturalOWL used, starting by adding sections to text planning (see Section 2.3.1.2). In lexicalization (see Section 2.3.2.1), we added adjectives and verbs as new types of entries to the lexicon, and we extended the sentence plans to support additional options (e.g., tense, polarity, etc.), number/person agreement between slots, slots that concatenate multiple property values of the message triple's object (Section 2.3.2.1) etc. Default sentence plans for more domain-dependent and modifier properties were also added, as well as the option to manually edit them. NL names (along with the option to declare classes and individuals as anonymous) were first introduced in version 2; version 1 had no NL names. In the absence of sentence plans and NL names, the current version of NaturalOWL attempts to automatically generate resources by tokenizing the OWL identifiers of the corresponding relations, classes, and individuals; version 1 was unable to generate text without fully authored linguistic resources. Finally, more aggregation rules, which take into account the presence of adjectives and modified properties, were implemented (see Section 2.3.2.2), as well as more options concerning interest scores in user modelling (Androutsopoulos et al., 2012). All the domain-dependent linguistic resources have been fully reworked in version 2 of NaturalOWL to be represented as OWL 2 ontologies themselves, facilitating their publication and sharing on the Web.

The experiments that examined the quality of the produced texts of version 2 of NaturalOWL with different ontologies, including the ablation experiments, are also novel; the quality of the texts of version 1 was never examined in such detail.

Finally, version 2 of NaturalOWL is accompanied by a new version of the NaturalOWL plug-in, which was developed during the work of this thesis (see Chapter 5).

2.6 Summary and Contributions of this Chapter

We provided a detailed description of NaturalOWL version 2, an open-source NLG system that was developed during the work of this thesis to produce English and Greek texts describing individuals or classes of OWL ontologies. Unlike simpler verbalizers, which typically express a single axiom at a time in controlled, often not entirely fluent English primarily for the benefit of domain experts, NaturalOWL aims to generate fluent and coherent multi-sentence texts for end-users in more than one languages.

We discussed the processing stages of our system, the optional domain-dependent generation resources of each stage, as well as particular NLG issues that arise when generating from OWL ontologies. We also presented trials we performed with ontologies concerning different domains to measure the effort required to construct the domain-dependent generation resources and the extent to which they improve the resulting texts, also comparing against a simpler OWL verbalizer that requires no domain-dependent generation resources and employs NLG methods to a lesser extent. The trials showed that the domain-dependent generation resources help NaturalOWL produce significantly better texts, and that the resources can be constructed with relatively light effort, compared to the effort that is typically needed to develop an OWL ontology, though several days may be needed. We also showed how the descriptions can be generated in more than one languages, again provided that appropriate resources are available. Our ablation trials concluded that natural language names, sentence plans, and (to a lesser extent) text plans have the greatest impact on text quality. In the next chapter, we propose methods to automatically or semi-automatically extract these domain-dependent generation resources from the Web. All the generation resources of NaturalOWL are also represented in OWL, facilitating their publication and sharing on the Web.

This chapter constitutes the first detailed discussion of a complete, general-purpose NLG system for OWL, while the new version of NaturalOWL is the first complete, publicly available NLG system for OWL, excluding much simpler ontology verbalizers.

Chapter 3

Extracting linguistic resources from the Web

3.1 Introduction

To make ontologies easier to understand for people unfamiliar with formal knowledge representations, several *ontology verbalizers* have been developed (Cregan et al., 2007; Kaljurand and Fuchs, 2007; Schwitter et al., 2008; Halaschek-Wiener et al., 2008; Schutte, 2009; Power and Third, 2010; Power, 2010; Schwitter, 2010a; Stevens et al., 2011; Liang et al., 2011b). Although verbalizers can be viewed as performing a kind of light natural language generation (NLG), they usually translate the axioms (in our case, OWL statements) of the ontology one by one to controlled, often not entirely fluent English statements, typically without considering the coherence of the resulting texts. By contrast, more elaborate NLG systems (Bontcheva, 2005; Androutsopoulos et al., 2007) can produce more fluent and coherent multi-sentence texts, but they need domain-specific linguistic resources. For example, NaturalOWL can produce the following description for the class `StEmilion` of the Wine Ontology. It needs, however: a *sentence plan* for each relation (e.g., `locatedIn`) of the ontology, a *natural lan-*

guage name for each class and individual (see Section 2.3.2.1), a text plan specifying the order in which relations should be expressed etc. Similar domain-specific linguistic resources are used in most concept-to-text systems (Reiter and Dale, 2000). Manually constructing resources of this kind, however, can be tedious and costly.

> St. Emilion is a kind of red, strong Bordeaux from the St. Emilion region. It is made from exactly one grape variety: Cabernet Sauvignon grapes.

Instead of requiring domain-specific linguistic resources, simpler verbalizers use the OWL identifiers of classes and individuals (e.g., `cabernetSauvignonGrape`) typically split into tokens as their natural language names, they express relations using phrases obtained by tokenizing the OWL identifiers of the relations (e.g., `hasColor`), they order the resulting sentences following the ordering of the corresponding OWL statements etc. Without domain-specific linguistic resources, NaturalOWL behaves like a simple verbalizer, producing the following lower quality text from the OWL statement above. A further limitation of using tokenized OWL identifiers is that non-English texts cannot be generated, since OWL identifiers are usually English-like.

> St Emilion is Bordeaux. St Emilion located in St Emilion Region. St Emilion has color Red. St Emilion has flavor Strong. St Emilion made from grape exactly 1: Cabernet Sauvignon Grape.

Previous experiments (see Section 2.4) indicate that the texts that NaturalOWL generates with domain-specific linguistic resources are perceived as significantly better than (i) those of SWAT, one of the best OWL verbalizers (Stevens et al., 2011; Williams et al., 2011), and (ii) those of NaturalOWL without domain-specific linguistic resources, with little or no difference between (i) and (ii). The largest difference in the perceived quality of the texts was reported to be due to the sentence plans, natural language names, and (to a lesser extent) text plans.

In this chapter, we present a method to automatically or semi-automatically extract from the Web the natural language names required by NaturalOWL for the individuals and classes of a given ontology. We also present a method to automatically or semi-automatically extract from the Web the sentence plans required by NaturalOWL for the relations of a given ontology. We do not examine how other types of domain-specific linguistic resources (e.g., text plans) can be automatically generated, leaving them for future work. The processing stages and linguistic resources of NaturalOWL are typical of NLG systems (Mellish et al., 2006b). Hence, we believe that our work is also applicable, at least in principle, to other NLG systems. Our methods may also be useful in simpler verbalizers, where the main concern seems to be to avoid the manual authoring of domain-specific linguistic resources. Experiments show that texts generated using linguistic resources extracted by our methods in a semi-automatic manner, with minimal human involvement, are perceived as being almost as good as texts generated using manually authored linguistic resources, and much better than texts produced by using tokenized OWL identifiers. By contrast, our methods do not perform sufficiently well in a fully automatic scenario, with no human involvement during the construction of the linguistic resources.

Sections 3.2 and 3.3 describe our methods to extract natural language names and sentence plans, respectively, from the Web. Section 3.4 presents our experimental results. Section 3.5 discusses related work. Section 3.6 concludes and and summarizes the research contribution of this chapter.

3.2 Our method to extract NL names from the Web

In this section we present our method to produce NL names. Given a target class or individual t that we want to produce an NL name for, our method begins by extracting noun phrases from the Web that are similar to the OWL identifier of t. The noun phrases

are ranked by aligning their words to the tokens of the original OWL identifier of t. The best noun phrases are then enhanced with linguistic annotations (e.g., POS tags, agreement constraints, number requirements), missing articles etc., turning them into NL names. The purpose of the method is to identify the best few (up to 5) candidate NL names for t. In a fully automatic scenario, the candidate NL name that the method considers best for t is then used. In a semi-automatic scenario, the few top (according to the method) NL names of t are shown to a human author, who picks the best one; this is much easier than manually authoring NL names.

3.2.1 Extracting noun phrases from the Web

We first collect all of the OWL statements of the ontology that describe t, the individual or class we want to produce an NL name for, and turn them into message triples $\langle S = t, R, O \rangle$, as when generating texts. For example, when considering the class $t =$ `KalinCellarsSemillon` of the Wine Ontology, one of the ontologies of our experiments, three of the resulting message triples are:

$$<\texttt{:KalinCellarsSemillon}, \texttt{isA}, \texttt{:Semillon}>$$

$$<\texttt{:KalinCellarsSemillon}, \texttt{:hasMaker}, \texttt{:KalinCellars}>$$

$$<\texttt{:KalinCellarsSemillon}, \texttt{:hasFlavor}, \texttt{:Strong}>$$

For each collected message triple $\langle S = t, R, O \rangle$, we then produce *tokName*(S) and *tokName*(O), where *tokName*(X) is the tokenized identifier of X.[1] From the three triples above, we obtain:

$$\textit{tokName}(\texttt{KalinCellarsSemillon}) = \text{“Kalin Cellars Semillon”},$$

$$\textit{tokName}(\texttt{Semillon}) = \text{“Semillon”}$$

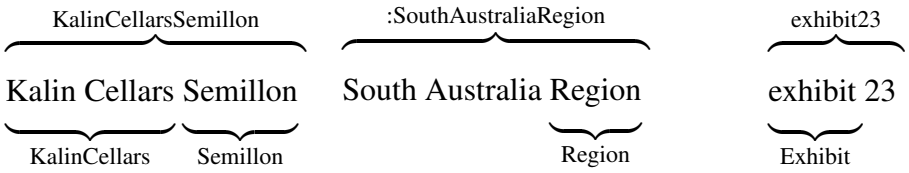
[1] Most OWL ontologies use identifiers written in CamelCase, or identifiers that can be easily broken into tokens at underscores, hyphens etc. If the ontology provides an `rdfs:label` for X, we use the tokens of the `rdfs:label` as *tokName*(X).

$$\textit{tokName}(\texttt{KalinCellars}) = \text{“Kalin Cellars”},$$

$$\textit{tokName}(\texttt{Strong}) = \text{“Strong”}$$

3.2.1.1 Shortening the tokenized identifiers

Subsequently, we attempt to shorten $\textit{tokName}(t)$, i.e., the tokenized identifier of the individual or class we wish to produce an NL name for, by removing any part (token sequence) of $\textit{tokName}(t)$ that is identical to the tokenized identifier of the O of any triple $\langle S = t, R, O \rangle$ that we collected for t. If the shortened tokenized identifier of t is the empty string or contains only numbers, t is marked as anonymous (see Section 2.3.2.3). In our example, where t = `KalinCellarsSemillon`, the tokenized identifier of t is initially $\textit{tokName}(t)$ = “Kalin Cellars Semillon”. We remove the part “Semillon”, because of the triple <`:KalinCellarsSemillon`, `isA`, `:Semillon`> and the fact that *tokName*(`Semillon`) = “Semillon”, as illustrated below. We also remove the remaining part “Kalin Cellars”, because of the triple <`:KalinCellarsSemillon`, `:hasMaker`, `:KalinCellars`> and the fact that *tokName*(`KalinCellars`) = “Kalin Cellars”. Hence, `KalinCellarsSemillon` is marked as anonymous.



This would have the effect of producing texts like the following, when asked to describe `KalinCellarsSemillon`:

> This is a strong, dry Semillon. It has a full body. It is made by Kalin Cellars.

instead of the following text, which repeats “Semillon” and “Kalin Cellars”:

> Kalin Cellars Semillon is a strong, dry Semillon. It has a full body. It is made by Kalin Cellars.

Similarly, if t = `SouthAustraliaRegion` and we have collected the following message triple, the tokenized identifier of t would be shortened from "South Australia Region" to "South Australia". We use *altTokName* to denote the resulting shortened tokenized identifiers.

$$<\texttt{:SouthAustraliaRegion}, \texttt{isA}, \texttt{:Region}>$$

tokName(`SouthAustraliaRegion`) = "South Australia Region", *tokName*(`Region`) = "Region"

altTokName(`SouthAustraliaRegion`) = "South Australia"

As a further example, if t = `exhibit23` and we have collected the following triple, the tokenized identifier of `exhibit23` would end up containing only numbers ("23"). Hence, `exhibit23` would be marked as anonymous.

$$<\texttt{:exhibit23}, \texttt{isA}, \texttt{:Exhibit}>$$

tokName(`exhibit23`) = "exhibit 23", *tokName*(`Exhibit`) = "exhibit"

3.2.1.2 Improving the tokenized identifiers by considering ancestor classes, numbers, brackets

We then collect the tokenized identifiers of all the ancestor classes of t, also taking into account equivalent classes; for example, if t has an equivalent class t', we also collect the tokenized identifiers of the ancestor classes of t', and similarly if an ancestor of t has an equivalent class. For t = `KalinCellarsSemillon`, we collect the following tokenized identifiers, because `Semillon`, `SemillonOrSauvignonBlanc`, and `Wine` are ancestors of t.[2]

[2]We consider only classes with OWL identifiers when traversing the hierarchy, ignoring classes constructed using OWL operators (e.g., class intersection) that have not been assigned OWL identifiers. In ontologies with multiple inheritance, the same tokenized identifiers of ancestors of t may be obtained by following different paths in the class hierarchy, but we remove duplicate tokenized identifiers. We discard the tokenized identifier of the most general class `owl:Thing`.

tokName(`Semillon`) = "Semillon", *tokName*(`SemillonOrSauvignonBlanc`) = "Semillon Or Sauvignon Blanc", *tokName*(`Wine`) = "Wine"

If the (unshortened) *tokName*(t) does not contain any of the collected tokenized identifiers of the ancestor classes of t, we create additional alternative tokenized identifiers for t, also denoted *altTokName*(t), by appending to *tokName*(t) the collected tokenized identifiers of the ancestor classes of t (again taking into account equivalent classes). For example, if t = `red` and `Color` is the parent class of t, we would also obtain "red color":

<`:red`, `isA`, `:Color`>

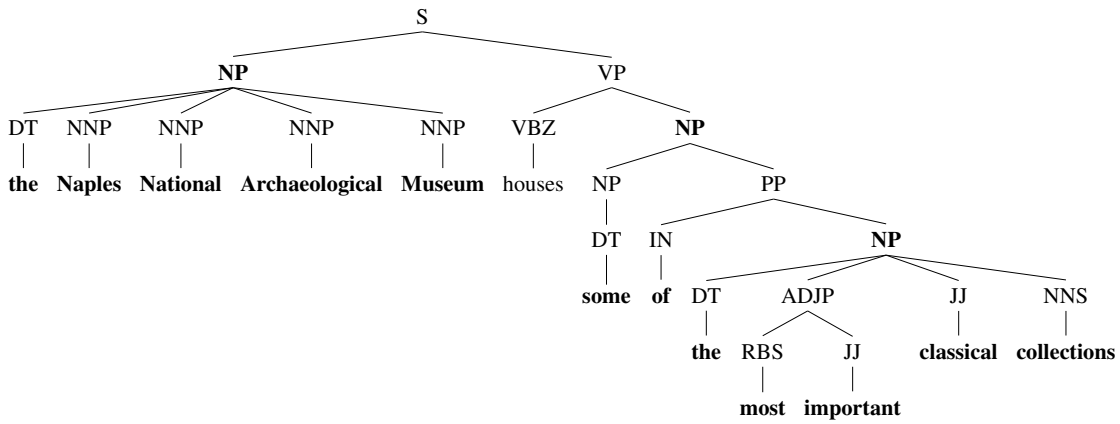
tokName(`red`) = "red", *tokName*(`Color`) = "color", *altTokName*(`red`) = "red color"

By contrast, if t = `KalinCellarsSemillon`, no *altTokName*(t) is produced from the ancestors of t, because *tokName*(t) = "Kalin Cellars Semillon" already contains *tokName*(`Semillon`) = "Semillon", and `Semillon` is the only known ancestor of `KalinCellarsSemillon`.

Furthermore, we create an additional *altTokName*(t) by removing all numbers from *tokName*(t); for example, from *tokName*(t) = "Semillon 2006" we obtain the shortened *altTokName*(t) ="Semillon". Lastly, if *tokName*(t) contains brackets, we create an *altTokName*(t) for each part outside and inside the brackets; for example, to use an example from a biomedical ontology we experiment with, from "gerbil (dessert rat)" we get "gerbil" and "dessert rat".

3.2.1.3 Retrieving Web pages, extracting and ranking noun phrases

Subsequently, we formulate a Boolean Web search query for *tokName*(t) (e.g., "South" `AND` "Australia" `AND` "Region") and each *altTokName*(t) (e.g., "South" `AND` "Australia"); recall that t is the individual or class we wish to produce an NL name for. If the search engine suggests corrections to any query, we use the corrected form of the query



Extracted noun phrases (NPs):

"**the Naples National Archaeological Museum**"

"**some of the most important classical collections**"

"**the most important classical collections**"

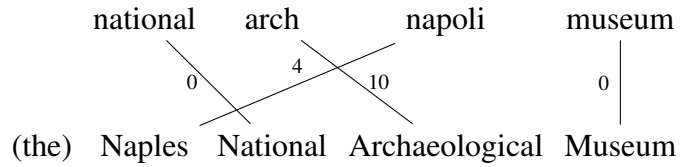
Figure 3.1: Parse tree of a retrieved sentence and its noun phrases.

as well. We convert the retrieved pages of all the queries to plain text documents and parse every sentence of the text, if any stemmed word of the sentence is the same as any stemmed word of any $tokName(t)$ or $altTokName(t)$.[3] We then extract the noun phrases (NPs) from every parsed sentence. For example, from the sentence "the Naples National Archaeological Museum houses some of the most important classical collections" we extract the NPs "the Naples National Archaeological Museum", "some of the most important classical collections", and "the most important classical collections" (Fig. 3.1).

For each one of the extracted NPs, we compute its similarity to $tokName(t)$ and each $altTokName(t)$. Let *np* be an extracted NP and let *name* be $tokName(t)$ or an $altTokName(t)$. To compute the similarity between *np* and *name*, we first compute

[3]We employ JSOUP (`http://jsoup.org/`) to obtain the plain texts from the Web pages, Porter's stemmer (`http://tartarus.org/martin/PorterStemmer/`), and also the Stanford parser (`http://nlp.stanford.edu/software/lex-parser.shtml`).

the character-based Levenshtein distance between each token of *np* and each token of *name*; we ignore upper/lower case differences, articles, and connectives (e.g., "and" 'or"), which are often omitted from OWL identifiers. In the following example, *np* = "the Naples National Archaeological Museum" (but the article "the" is ignored) and *name* = "national arch napoli museum"; the *name* is an $altTokName(t)$ produced by appending to $tokName(t)$ the tokenized identifier of the parent class (`Museum`) of t (Section 3.2.1.2). The Levenshtein distance between the "national" and "National" is 0 (upper/lower case differences are ignored), whereas the distance between "napoli" and "Naples" is 4; a character deletion or insertion costs 1, and a replacement costs 2.



We then form pairs of aligned tokens $\langle t_{name}, t_{np} \rangle$, where t_{name}, t_{np} are tokens from *name*, *np*, respectively, such that each token of *name* is aligned to at most one token of *np* and vice versa, and any other, not formed pair $\langle t'_{name}, t'_{np} \rangle$ would have a Levenshtein distance (between t'_{name}, t'_{np}) larger or equal to the minimum Levensthein distance of the formed pairs.[4] In our example, the pairs of aligned tokens are <"national", "National">, <"arch", "Archaeological">, <"napoli", "Naples">, <"museum", "Museum">.

The similarity between *np* and *name* is then computed as follows, where A is the set of aligned token pairs, $Levenshtein(a)$ is the Levenshtein distance between the t_{name} and t_{np} of pair a normalized (per token pair) to $[0,1]$, and $|np|$, $|name|$ are the lengths (in tokens) of *np* and *name*, respectively.

[4]In our experiments, we actually formed the pairs in a greedy manner, by computing the Levenshtein distances of all the possible pairs and iteratively selecting the pair with the smallest Levenshtein distance whose elements did not occur in any other already selected pair. A non-greedy search (e.g., using Integer Linear Programming) can also be used, but it makes little difference in practice.

$$similarity(np, name) = \frac{\sum_{a \in A}(1 - Levenshtein(a))}{max\{|np|, |name|\}} \quad (3.1)$$

For each extracted NP of t, we compute its similarity to every possible *name*, i.e., $tokName(t)$ or $altTokName(t)$, as discussed above, and we assign to the NP a score equal to the largest of these similarities. Finally, we rank the extracted NPs of t by decreasing score. If two NPs have the same score, we rank higher the NP with the fewest crossed edges in its best alignment with a *name*. If two NPs still cannot be distinguished, we rank them by decreasing frequency in the parsed sentences of t; and if their frequencies are equal, we rank them randomly.

3.2.2 Turning the extracted noun phrases into NL names

The extracted NPs of the previous section are not yet NL names, because they lack the linguistic annotations that NaturalOWL requires (e.g., POS tags, agreement constraints, number requirements); they may also lack appropriate articles. To convert an NP to an NL name, we first obtain the POS tags of its words from the parse tree of the sentence the NP was extracted from.[5] For example, the NP "the Red Wine" becomes:

$$\text{the}_{\text{POS}=DT}\ \text{Red}_{\text{POS}=JJ}\ \text{Wine}_{\text{POS}=NN}$$

For every noun, adjective, article, and preposition, we create a corresponding slot in the NL name; all the other words of the NP become slots filled in with the words as fixed strings (Section 2.3.2.1). For nouns and adjectives, we first consider any lexicon entries already available in our linguistic resources(see Section 2.3.2.1).[6] If a noun or adjective is not already defined as a lexicon entry, we employ SIMPLENLG (Gatt and Reiter, 2009) to obtain its various forms that are required to construct an appropriate lexicon entry;

[5] If an NP has been extracted from multiple sentences and their parse trees provide different POS tag assignments to the words of the NP, we create a separate NL name for each POS tag assignment.

[6] In our experiments of Section 3.4, we assume that no lexicon entries are available.

subsequently we create a corresponding slot referring to that lexicon entry. Specifically for nouns, a named-entity recognizer (NER) and an on-line dictionary are employed to detect which nouns refer to persons and locations.[7] The genders of these nouns are determined using the on-line dictionary, when possible, or defaults otherwise (e.g., the default for person nouns is a 'person' pseudo-gender, which leads to "he/she" or "they" when generating a pronoun). Nouns that do not refer to persons and locations are marked as neuter. Since all NPs are extracted from Web pages, there is a high risk of wrong capitalization (e.g. "The Red Wine" or "the RED wine"). For each word of the NL name, we pick the capitalization that is most frequent in the retrieved texts of t, the individual or class we generate the NL name for. Hence, the NP "the Red Wines" becomes:

$$[]^{1}_{article,def}\ [\text{red}]^{2}_{adj}\ [\text{wine}]^{3}_{noun,sing,neut}$$

The NL name above instructs NaturalOWL to produce a definite article, followed by the adjective "red", followed by the neuter noun "wine" in singular.

A dependency parser is then used to identify the head noun or adjective of each NL name (see Section 2.3.2.1) and to obtain agreement information.[8] Adjectives are required to agree with the nouns they modify, and the same applies to articles and nouns. At this stage, the NP "the Red Wines" will have become:

$$[]^{1}_{article,def,agr=3}\ [\text{red}]^{2}_{adj}\ [\text{wine}]^{3}_{headnoun,sing,neut}$$

The final stage considers the main article (or, more generally, determiner) of the NL name, i.e., the article (or determiner) that agrees with the head noun or head adjective

[7] We use the Stanford NER (`http://nlp.stanford.edu/`), and Wiktionary (`http://en.wiktionary.org/`).

[8] We use the Stanford parser (`http://nlp.stanford.edu/`), which can also produce dependency trees. The parser is applied to the sentences the NPs were extracted from. If multiple parses are obtained for the NP an NL name is based on, we keep the most frequent parse.

(e.g., "a" in the case of "a traditional wine from the Piemonte Region"). Although the NL name may already include a main article, it is not necessarily an appropriate one. For example, it would be inappropriate to use a definite article in "*The* red wine is a kind of wine with red color", when describing the class of red wines. We modify the NL name to use an indefinite article if the NL name refers to a class, and a definite article if it refers to an individual (e.g., "the South Australia region").[9] The article is omitted if the head is an adjective (e.g., "strong"), or in plural (e.g., "Semillon grapes"), or if the entire NL name (excluding the article, if present) is a proper name (e.g., "South Australia") or a mass noun phrase (without article, e.g., "gold"). Before inserting or modifying the main article, we also remove any demonstratives (e.g., "*this* statue") or other non-article determiners (e.g., "some", "all") from the beginning of the NL name. In our example, the NL name is to be used to refer to the class `RedWine`, so the final NL name is the following, which would lead to sentences like "*A* red wine is a kind of wine with red color".

$$[]^{1}_{article,indef,agr=3}\ [\text{red}]^{2}_{adj}\ [\text{wine}]^{3}_{headnoun,sing,neut}$$

Recall that NaturalOWL can automatically adjust NL names when generating texts (see Section 2.3.2.1). For example, NaturalOWL has been augmented to automatically generate comparisons to previously encountered individuals or classes (Karakatsiotis, 2007), as in "Unlike *the previous red wines* that you have seen, this one is from France"; here NaturalOWL would use a definite article and it would turn the head noun of the NL name to plural, also adding the adjective "previous".

The resulting NL names are ranked by the scores of the NPs they were obtained from (Section 3.2.1.3).

[9] This arrangement works well in the ontologies we have experimented with, but it may have to be modified for other ontologies, for example if kinds of entities are modeled as individuals, rather than classes.

3.2.3 Inferring interest scores from NL names

The reader may have already noticed that the sentence "A red wine is a kind of wine with red color" that we used above sounds redundant. Indeed, some message triples lead to sentences that sound redundant, because they report relations that are obvious (to humans) from the NL names of the involved individuals or classes. In our particular example, the sentence reports the following two message triples.

<:RedWine, isA, :Wine>, <:RedWine, :hasColor, :Red>

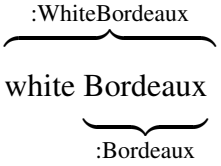
Expressed separately, the two triples would lead to the sentences "A red wine is a kind of wine" and "A red wine has red color", but NaturalOWL aggregates them into a single sentence. The "red color" derives from an *altTokName* of `Red` obtained by considering the parent class `Color` of `Red` (Section 3.2.1.2). It is obvious that a red wine is a wine with red color and, hence, the two triples above should not be expressed. Similarly, the following triple leads to the sentence "A white Bordeaux wine is a kind of Bordeaux", which again seems redundant.

<:WhiteBordeaux, isA, :Bordeaux>

In NaturalOWL we can assign a zero interest score to a triple in an appropriate User Model to avoid expressing that triple (see Section 2.3.1.1). Manually assigning interest scores, however, can be tedious. Hence, we aimed to automatically assign zero scores to triples like the ones above, which report relations that are obvious from the NL names. To identify triples of this kind, we follow a procedure similar to the one used to identify individuals or classes that should be anonymous (Section 3.2.1.1). For each $\langle S, R, O \rangle$ triple that involves the individual or class S being described, we examine the NL names that will be used to express S and O respectively.[10] If all the (lemmatized) words of the phrase produced by the NL name of O (e.g., "a Bordeaux") (excluding articles) appear

[10] If there are multiple available NL names for either S or O, NaturalOWL chooses one randomly.

in the phrase produced by the NL name of S (e.g., "a white Bordeaux"), as illustrated below, we assign a zero interest score to $\langle S, R, O \rangle$.



3.3 Our method to automatically extract sentence plans from the Web

We now present our method to produce sentence plans for NaturalOWL. Our method first extracts slotted string templates (e.g., "X is made from Y") from the Web using seeds (intuitively, values of X and Y) obtained from the ontology. Similarly to our method for NL name extraction, it then enhances the templates by adding linguistic annotations (e.g., POS tags, agreement constraints, voice, tense) and missing components (e.g., auxiliary verbs) turning the templates into candidate sentence plans. The candidate sentence plans are then scored by a Maximum Entropy classifier that examines features of the sentence plans themselves, but also features of the seeds and the texts that the sentence plans were extracted from.[11] As with the method of Section 3.2, the purpose of the classifier is to identify the best few (again up to 5) candidate sentence plans for each relation. In a fully automatic scenario, the sentence plan that the classifier considers best for each relation is used. In a semi-automatic scenario, the few top sentence plans of each relation are shown to a human author, who picks the best one.

[11] We use the Stanford classifier; `http://nlp.stanford.edu/software/classifier.shtml`.

3.3.1 Extracting templates from the Web

For each relation R that we want to generate a sentence plan for, our method first obtains the OWL statements of the ontology that involve the relation and turns them into message triples $\langle S,R,O\rangle$, as when generating texts.[12] For example, if the relation is `madeFrom`, two of the resulting triples may be:

<:StEmilion, :madeFrom, :cabernetSauvignonGrape>

<:Semillon, :madeFrom, :SemillonGrape>

To these triples, we add more by replacing the S, O, or both of each originally obtained triple by their classes (if S or O are individuals), their parent classes, or their equivalent classes. From <:StEmilion, :madeFrom, :cabernetSauvignonGrape>, for example, we also obtain the following three triples, because `Wine` is a parent class of `StEmilion`, and `Grape` is a parent class of `cabernetSauvignonGrape`.

<:Wine, :madeFrom, :cabernetSauvignonGrape>

<:StEmilion, :madeFrom, :Grape>, <:Wine, :madeFrom, :Grape>

From the triple <:Semillon, :madeFrom, :SemillonGrape> we obtain the same additional triples, because `Wine` and `Grape` are also parent classes of `Semillon` and `SemillonGrape`, respectively, but we remove duplicate triples.

Each $\langle S,R,O\rangle$ triple is then replaced by a pair $\langle seedName(S), seedName(O)\rangle$, where $seedName(S)$ is a word sequence generated by the NL name of S, and similarly for O. We assume that the NL names are manually authored (one per individual or class), or that they are generated by our method of Section 3.2. In the latter case, we keep only one NL name per individual or class, the one selected by the human author (in a semi-automatic setting of NL name generation) or the top ranked NL name (in a fully automatic setting). The five triples above become the following pairs. We call pairs

[12] We also obtain triples from OWL statements for each inverse relation of R in the ontology.

of this kind *seed name pairs*, and their elements *seed names*. If a seed name results from a class, parent-class, or an equivalent class of the original *S* or *O*, we consider it a *secondary seed name*.

<“St. Emilion”, “Cabernet Sauvignon grape”>, <“Semillon”, “Semillon grape”>

<“wine”, “Cabernet Sauvignon grape”>, <“St. Emilion”, “grape”>, < “wine”, “grape”>

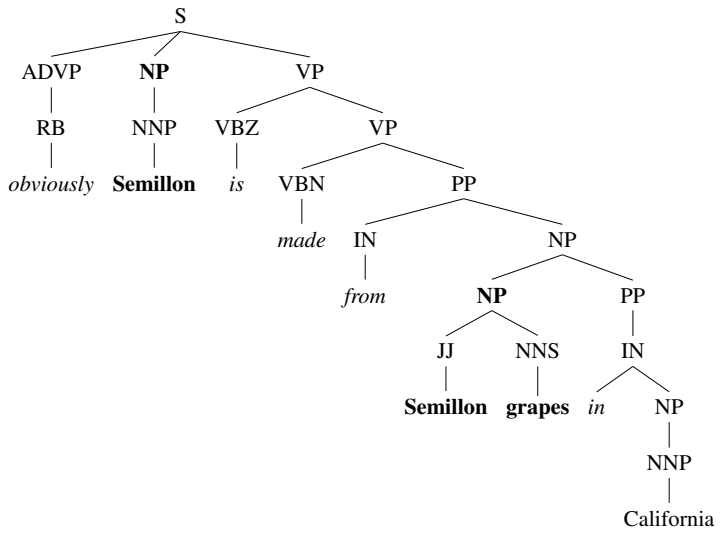
We then retrieve Web pages using the seed name pairs (of the relation that we want to generate a sentence plan for) as queries. For each seed name pair, we use the conjunction of its seed names (e.g., “St. Emilion” AND “Cabernet Sauvignon grape”) as a Boolean query.[13] We convert all the retrieved pages (of all the seed name pairs) to plain text documents and parse every sentence of the retrieved documents, if at least one stemmed word from both seed names of a seed name pair is the same as any stemmed word of the sentence.[14] We then keep every parsed sentence that contains at least two NPs matching a seed name pair. For example, the sentence “obviously Semillon is made from Semillon grapes in California” contains the NPs “Semillon” and “Semillon grapes” that match the seed name pair <“Semillon”, “Semillon grape”> (Fig. 3.2). Two NPs of a sentence match a seed name pair if the similarity between any of the two NPs and any of the two seed names (e.g., the first NP and the second seed name) is above a threshold T and the similarity between the other NP and the other seed name is also above T. The similarity between an NP and a seed name is computed as their weighted cosine similarity, with $tf \cdot idf$ weights, applied to stemmed NPs and seed names, ignoring stopwords.[15] The *tf* of a word of the NP or seed name is the frequency (usually 0 or 1) of the word in the NP or seed name, respectively; the *idf* is the inverse document frequency

[13] In this method, we do not consider corrections suggested by the search engine, since the NL names are assumed to be correct.

[14] Again, we employ JSOUP (`http://jsoup.org/`) to obtain the plain texts from the Web pages, Porter’s stemmer (`http://tartarus.org/martin/PorterStemmer/`), and the Stanford parser (`http://nlp.stanford.edu/`).

[15] We also remove accents, and replace all numeric tokens by a particular pseudo-token.

of the word in all the retrieved documents of the relation. Note that some NPs and seed names may contain several words (e.g., "traditional wine from the Piemonte region"). We call NP *anchor pair* any pair that contains two NPs of a parsed sentence that match a seed name pair (in the order they match the seed name pair), and we call NP *anchors* the elements of an NP anchor pair.



Seed name pairs:

<"St. Emilion", "Cabernet Sauvignon grape">

<"**Semillon**", "**Semillon grape**">

<"wine", "Cabernet Sauvignon grape">

<"St. Emilion", "grape">

< "wine", "grape">

NP anchor pair:

<"**Semillon**", "**Semillon grapes**">

Figure 3.2: Parse tree of a retrieved sentence and its single NP anchor pair.

From every parsed sentence that contains an NP anchor pair, we produce a slotted string template by replacing the first NP anchor by *S*, the second NP anchor by *O*, including between *S* and *O* in the template the words of the sentence that were between the two NP anchors, and discarding the other words of the sentence. In the example of Fig. 3.2, we would obtain the template "*S* is made from *O*". Multiple templates may be

extracted from the same sentence, if a sentence contains more than one NP anchor pairs; and the same template may be extracted from multiple sentences, possibly retrieved by different seed name pairs. We retain only the templates that were extracted from at least two different sentences. We then produce additional templates by increasingly extending the retained ones up to the boundaries of (all) the sentences they were extracted from. In the example of Fig. 3.2, assuming that the template "*S* is made from *O*" has been retained, we would also produce the following templates.

obviously *S* is made from *O* obviously *S* is made from *O* in

S is made from *O* in *S* is made from *O* in California

obviously *S* is made from *O* in California

Again, we discard extended templates that did not result from at least two sentences.

3.3.2 Turning the templates into candidate sentence plans

The templates produced by the process of the previous section (e.g., "*S* is made from *O*") are not yet sentence plans, because they lack the linguistic annotations that NaturalOWL requires its sentence plans to carry (e.g., POS tags, agreement constraints, voice, tense, cases), and they may also not correspond to well-formed sentences (e.g., they may lack verbs). The conversion of a template to a (candidate) sentence plan is similar to the conversion of an NP to an NL name of Section 3.2.2. We start by obtaining POS tags from the parse trees of the sentences the template was obtained from. Recall that a template may have been extracted from multiple sentences. We obtain a POS tag sequence for the words of the template from each one of the sentences the template was extracted from, and we keep the most frequent POS tag sequence. We ignore the POS tags of the anchor NPs, which become *S* and *O* in the template. For example, the template "*S* is made from *O*" becomes:

$$S\ \text{is}_{\text{POS}=VBZ}\ \text{made}_{\text{POS}=VBN}\ \text{from}_{\text{POS}=IN}\ O$$

For every verb form (e.g., "is made"), noun, adjective, and preposition, we create a corresponding slot in the sentence plan. For verbs, nouns, and adjectives, a reference to an appropriate lexicon entry is used in the slot as in Section 3.2.2; each verb slot is also annotated with the voice and tense of the corresponding verb form in the template.[16] If a negation expression (e.g., "not", "aren't") is used with the verb form in the template, the negation expression is not included as a separate slot in the sentence plan, but the polarity of the verb slot is marked as negative; otherwise the polarity is positive. We determine the genders and capitalizations of nouns (and proper names) using the same techniques as in Section 3.2.2. The *S* and *O* are also replaced by slots requiring referring expressions. For example, the template "*S* is made from *O*" will have now become:

$$[ref(S)]^1\ [\text{make}]^2_{verb,passive,present,polarity=+}\ [\text{from}]^3_{prep}\ [ref(O)]^4$$

Agreement and case information is obtained using a dependency parser.[17] The parser is applied to the sentences the templates were extracted from, keeping the most frequent parse per template. Referring expressions obtained from NP anchors that were verb subjects are marked with nominative case, and they are required to agree with their verbs. Referring expressions corresponding to verb objects or preposition complements are marked with accusative case (e.g., "from *him*"). Referring expressions corresponding to NP anchors with head nouns in possessive form (e.g., "Piemonte's") are marked with possessive case. In our example, we obtain:

$$[ref(S)]^1_{case=nom}\ [\text{make}]^2_{verb,passive,present,agr=1,polarity=+}\ [\text{from}]^3_{prep}\ [ref(O)]^4_{case=acc}$$

Any remaining words of the template that have not been replaced by slots (e.g., "obviously" in "obviously *S* is made from *O*") are turned into fixed string slots. Subsequently, any sentence plan that has only two slots, starts with a verb, or contains no

[16] Again, we employ SIMPLENLG (Gatt and Reiter, 2009) to obtain the base forms, voices, and tenses.

[17] Again, we use the Stanford parser (`http://nlp.stanford.edu/`).

verb, is discarded, because sentence plans of these kinds tend to produce ungrammatical sentences. Also, if a sentence plan contains a single verb in the past participle, in agreement with either *S* or *O*, followed by a preposition (e.g. "*S* made in *O*"), we insert an auxiliary verb to turn the verb form into present passive (e.g., "*S* is made in *O*"); in domains other than those of our experiments, a past passive may be more appropriate ("*S* was made in *O*). Similarly, if a single verb appears in the present participle (e.g. "*S* making *O*"), we insert an auxiliary verb to obtain a present continuous form. Both cases are illustrated below.

S made in *O* $\Rightarrow$ *S* <u>is made</u> in *O* *S* making *O* $\Rightarrow$ *S* <u>is</u> making *O*

Lastly, we filter the remaining sentence plans through a Web search engine. For this step, we replace referring expression slots by wildcards, we generate the rest of the sentence that corresponds to the other slots (e.g., "* is made from *") and we do a phrase search. If no results are returned, the sentence plan is discarded.

3.3.3 Applying a Maximum Entropy classifier to the candidate sentence plans

The retained sentence plans of the previous section are then treated as candidate sentence plans, and they are scored using a Maximum Entropy (MaxEnt) classifier to pick the best ones. The classifier views each candidate sentence plan *p* for a relation *R* as a vector of 251 features, and attempts to estimate the probability that *p* is a good sentence plan (positive class) for *R* or not (negative class). The 251 features provide information about the particular candidate sentence plan *p* being scored, but also about the templates, seed name pairs, and NP anchor pairs of *p*, meaning the templates that *p* was obtained from (Section 3.3.2), and the seed name pairs and NP anchor pairs (Fig. 3.2) that matched to produce the templates of *p* (Section 3.3.1).

3.3.3.1 Productivity features

The *productivity of the i-th seed name pair* $\langle n_{i,1}, n_{i,2} \rangle$ (e.g., $<n_{i,1}$ ="Semillon", $n_{i,2}$ = "Semillon grape"$>$) of a relation R (e.g., $R =$ `madeFrom`) is defined as follows:

$$productivity(\langle n_{i,1}, n_{i,2} \rangle | R) = \frac{hits(\langle n_{i,1}, n_{i,2} \rangle | R)}{\sum_{j=1}^{N} hits(\langle n_{j,1}, n_{j,2} \rangle | R)} \tag{3.2}$$

where: $hits(\langle n_{i,1}, n_{i,2} \rangle | R)$ is the number of times $\langle n_{i,1}, n_{i,2} \rangle$ matched any NP anchor pair of the parsed sentences of R, counting only matches that contributed to the extraction (Section 3.3.1) of *any* template of R; N is the total number of seed name pairs of R; and $\langle n_{j,1}, n_{j,2} \rangle$ is the j-th seed name pair of R.[18] The intuition behind $productivity(\langle n_{i,1}, n_{i,2} \rangle | R)$ is that seed name pairs that match NP anchor pairs of many sentences of R are more likely to be indicative of R. When using the MaxEnt classifier to score a sentence plan p for a relation R, we compute the $productivity(\langle n_{i,1}, n_{i,2} \rangle | R)$ of all the seed name pairs $\langle n_{i,1}, n_{i,2} \rangle$ of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five features of p.

The *productivity of a seed name* n_1 (considered on its own) that occurs as the first element of at least one seed name pair $\langle n_{i,1}, n_{i,2} \rangle = \langle n_1, n_{i,2} \rangle$ of a relation R is defined as follows:

$$productivity(n_1 | R) = \frac{\sum_{i=1}^{N_2} hits(\langle n_1, n_{i,2} \rangle | R)}{\sum_{j=1}^{N} hits(\langle n_{j,1}, n_{j,2} \rangle | R)} \tag{3.3}$$

where: N_2 is the number of seed name pairs $\langle n_1, n_{i,2} \rangle$ of R that have n_1 as their first element; $hits(\langle n_1, n_{i,2} \rangle | R)$ is the number of times n_1 (as part of a seed name pair $\langle n_1, n_{i,2} \rangle$ of R) matched any element of any NP anchor pair of the parsed sentences of R and contributed to the extraction of *any* template of R; N and $hits(\langle n_{j,1}, n_{j,2} \rangle | R)$ are as in

[18] In all the equations that use the function $hits(\langle n_1, n_2 \rangle | R)$, we actually multiply by $\frac{1}{2}$ the value returned by the function if exactly one of n_1, n_2 is a secondary seed name (Section 3.3.1), and we multiply by $\frac{1}{4}$ if both n_1, n_2 are secondary seed names. The same applies to the function $hits(\langle n_1, n_2 \rangle, t | R)$ of Eq. 3.8 and the function $hits(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle, t | R)$ of Eq. 3.9 below.

Eq. 3.2. Again, when using the classifier to score a sentence plan p for a relation R, we calculate the *productivity*$(n_1|R)$ values of all the distinct (duplicates considered once) seed names n_1 that occur as first elements in the seed name pairs of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five more features of p. We define similarly *productivity*$(n_2|R)$ for a seed name n_2 that occurs as the *second* element in any seed name pair $\langle n_{i,1}, n_2 \rangle$ of R, obtaining five more features for p.

Similarly to Eq. 3.2, we define the *productivity of the i-th NP anchor pair* $\langle a_{i,1}, a_{i,2} \rangle$ (e.g., $<a_{1,1}$ ="Semillon", $a_{1,2}$ = "Semillon grapes"> in Fig. 3.2) of a relation R as follows:

$$productivity(\langle a_{i,1}, a_{i,2} \rangle |R) = \frac{hits(\langle a_{i,1}, a_{i,2} \rangle |R)}{\sum_{j=1}^{A} hits(\langle a_{j,1}, a_{j,2} \rangle |R)} \tag{3.4}$$

where: $hits(\langle a_{i,1}, a_{i,2} \rangle |R)$ is the number of times a seed name pair of R matched the NP anchor pair $\langle a_{i,1}, a_{i,2} \rangle$ in the parsed sentences of R and contributed to the extraction of *any* template of R; and A is the total number of NP anchor pairs of R.[19] As with *productivity*$(\langle n_{i,1}, n_{i,2} \rangle |R)$, the intuition behind *productivity*$(\langle a_{i,1}, a_{i,2} \rangle |R)$ is that NP anchor pairs that match many seed name pairs of R are more indicative of R. When using the classifier to score a sentence plan p for a particular relation R, we compute the *productivity*$(\langle a_{i,1}, a_{i,2} \rangle |R)$ of all the NP anchor pairs of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these scores as five additional features of p.

Similarly to Eq. 3.3, the *productivity of an NP anchor* a_1 (considered on its own) that occurs as the first element of at least one NP anchor pair $\langle a_{i,1}, a_{i,2} \rangle = \langle a_1, a_{i,2} \rangle$ of R is defined as follows:

[19] In all the equations that use the function $hits(\langle a_1, a_2 \rangle |R)$, we actually multiply by $\frac{1}{2}$ any match of $\langle a_1, a_2 \rangle$ with a seed name pair $\langle n_1, n_2 \rangle$ if exactly one of n_1, n_2 is a secondary seed name, and $\frac{1}{4}$ if both n_1, n_2 are secondary seed names.

$$productivity(a_1|R) = \frac{\sum_{i=1}^{A_2} hits(\langle a_1, a_{i,2} \rangle |R)}{\sum_{j=1}^{A} hits(\langle a_{j,1}, a_{j,2} \rangle |R)} \quad (3.5)$$

where: A_2 is the number of NP anchor pairs $\langle a_1, a_{i,2} \rangle$ of R that have a_1 as their first element; $hits(\langle a_1, a_{i,2} \rangle |R)$ is the number of times a_1 (as part of an NP anchor pair $\langle a_1, a_{i,2} \rangle$ of R) matched any element of any seed name pair of R and contributed to the extraction of *any* template of R; and A, $hits(\langle a_{j,1}, a_{j,2} \rangle |R)$ are as in Eq. 3.4. Again, when using the classifier to score a sentence plan p for a relation R, we calculate the $productivity(a_1|R)$ values of all the distinct (duplicates considered once) NP anchors a_1 that occur as first elements in the NP anchor pairs of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five more features of p. We define similarly $productivity(a_2|R)$ for a seed name a_2 that occurs as the *second* element in any NP anchor pair $\langle a_{i,1}, a_2 \rangle$ of R, obtaining five more features for p.

The *productivity of a template t* (e.g., "*S* is made from *O*") of a relation R is defined as follows:

$$productivity(t|R) = \frac{hits(t|R)}{\sum_{k=1}^{T} hits(t_k|R)} \quad (3.6)$$

where: $hits(t|R)$ is the number of times the *particular* template t was extracted from any of the parsed sentences of R; T is the total number of templates of R; and t_k is the k-th template of R. The intuition is that templates that are produced more often for R are more indicative of R. When using the classifier to score a sentence plan p for a relation R, we calculate the $productivity(t|R)$ values of all the templates t of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five more features of p.

The *productivity of a parsed sentence s* (e.g., "obviously Semillon is made from Semillon grapes in California") of a relation R is defined as follows:

$$productivity(s|R) = \frac{hits(s|R)}{\sum_{l=1}^{L} hits(s_l|R)} \tag{3.7}$$

where: $hits(s|R)$ is the number of times any template of R was extracted from the *particular* parsed sentence s; L is the total number of parsed sentences of R; and s_l is the l-th parsed sentence of R. The intuition is that sentences that produce more templates for R are more indicative of R. When using the classifier to score a sentence plan p for R, we calculate the $productivity(s|R)$ values of all the parsed sentences s of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five more features of p.

The *joint productivity* of a seed name pair $\langle n_1, n_2 \rangle$ and a template t of a relation R is defined as follows:

$$productivity(\langle n_1, n_2 \rangle, t|R) = \frac{hits(\langle n_1, n_2 \rangle, t|R)}{\sum_{j=1}^{N} \sum_{k=1}^{T} hits(\langle n_{j,1}, n_{j,2} \rangle, t_k|R)} \tag{3.8}$$

where: $hits(\langle n_1, n_2 \rangle, t|R)$ is the number of times the *particular* seed name pair $\langle n_1, n_2 \rangle$ matched any NP anchor pair of the parsed sentences of R and contributed to the extraction of the *particular* template t; and N, T are again the total numbers of seed name pairs and templates, respectively, of R. When using the classifier to score a sentence plan p for a relation R, we calculate the $productivity(\langle n_1, n_2 \rangle, t|R)$ of all the combinations of seed name pairs $\langle n_1, n_2 \rangle$ and templates t that led to p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five more features of p. We define very similarly $productivity(n_1, t|R)$, $productivity(n_2, t|R)$, $productivity(\langle a_1, a_2 \rangle, t|R)$, $productivity(a_1, t|R)$, $productivity(a_2, t|R)$, $productivity(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle |R)$, $productivity(n_1, a_1|R)$, $productivity(n_2, a_2|R)$, obtaining five additional features of p from each one. We also define:

$$productivity(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle, t|R) = \frac{hits(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle, t|R)}{\sum_{i=1}^{N} \sum_{j=1}^{A} \sum_{k=1}^{T} hits(\langle n_{i,1}, n_{i,2} \rangle, \langle a_{j,1}, a_{j,2} \rangle, t_k|R)} \tag{3.9}$$

where: $hits(\langle n_1, n_2\rangle, \langle a_1, a_2\rangle, t|R)$ is the number of times the seed name pair $\langle n_1, n_2\rangle$ matched the NP anchor pair $\langle a_1, a_2\rangle$ in a parsed sentence of R and contributed to the extraction of the template t; and N, A, T are the total numbers of seed name pairs, NP anchor pairs, and templates of R. We define very similarly $productivity(n_1, a_1, t|R)$ and $productivity(n_2, a_2, t|R)$,obtaining five features of p from each one.

3.3.3.2 Prominence features

For each one of the *productivity* versions of Section 3.3.3.1, we define a corresponding variant, called *prominence*. For example, based on the productivity of a seed name pair $\langle n_{i,1}, n_{i,2}\rangle$ of a relation R (Eq. 3.2, repeated as Eq. 3.10),

$$productivity(\langle n_{i,1}, n_{i,2}\rangle|R) = \frac{hits(\langle n_{i,1}, n_{i,2}\rangle|R)}{\sum_{j=1}^{N} hits(\langle n_{j,1}, n_{j,2}\rangle|R)} \tag{3.10}$$

we define the *prominence of a candidate sentence plan p* with respect to the *seed name pairs* of R as follows:

$$prominence_{seed\_name\_pairs}(p|R) = \frac{\sum_{i=1}^{N} 1\{hits(\langle n_{i,1}, n_{i,2}\rangle|p, R) > 0\}}{\sum_{j=1}^{N} 1\{hits(\langle n_{j,1}, n_{j,2}\rangle|R) > 0\}} \tag{3.11}$$

where: the notation $1\{\xi\}$ denotes a value of 1 if condition ξ holds and 0 otherwise; $hits(\langle n_{i,1}, n_{i,2}\rangle|p, R)$ (in the numerator) is the number of times $\langle n_{i,1}, n_{i,2}\rangle$ matched any NP anchor pair of the parsed sentences of R, counting only matches that contributed to the extraction of a template of R that led to the *particular sentence plan p*; by contrast, $hits(\langle n_{j,1}, n_{j,2}\rangle|R)$ (in the denominator) is the number of times $\langle n_{j,1}, n_{j,2}\rangle$ matched any NP anchor pair of the parsed sentences of R, counting only matches that contributed to the extraction of *any template* of R; and N is the total number of seed name pairs of R. In other words, we count how many (distinct) seed name pairs of R produced p, dividing by the number of (distinct) seed name pairs of R that produced at least one

template of R. The intuition is that the more (distinct) seed name pairs of R lead to the candidate sentence plan p, the better p is.

In a similar manner, we define $prominence_{anchor\_pairs}(p|R)$ based on Eq. 3.4, and similarly for all the other productivity versions of Section 3.3.3.1. We obtain one feature for the candidate sentence plan p from each prominence variant, i.e., we do not compute any maximum, minimum, average, sum, and standard deviation values, unlike the productivity versions, which lead to five features each.

3.3.3.3 PMI features

To estimate the extent to which two seed names n_1, n_2 of a relation R co-occur when they match NP anchors to produce templates of R, we use a normalized version of Pointwise Mutual Information (PMI), defined below:

$$\text{PMI}(\langle n_1, n_2 \rangle | R) = \frac{1}{-\log productivity(\langle n_1, n_2 \rangle | R)} \cdot \log \frac{productivity(\langle n_1, n_2 \rangle | R)}{productivity(n_1|R) \cdot productivity(n_2|R)} \quad (3.12)$$

The second factor of the right-hand side of Eq. 3.12 is the standard PMI definition, using productivity scores instead of probabilities. The first factor normalizes the PMI scores to $[-1,1]$ (-1 if n_1, n_2 never co-occur when producing templates of R, 0 if they are independent, 1 if they always co-occur). Intuitively, if n_1, n_2 co-occur frequently when they produce templates of R, they are strongly connected and, hence, they are more indicative of R. Again, when using the classifier to score a sentence plan p for a relation R, we calculate $\text{PMI}(\langle n_1, n_2 \rangle | R)$ for all the seed name pairs $\langle n_1, n_2 \rangle$ of R, and use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these PMI scores as five more features of p. We define similarly $\text{PMI}(\langle n_1, n_2 \rangle, t|R)$, $\text{PMI}(n_1, t|R)$, $\text{PMI}(n_2, t|R)$, $\text{PMI}(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle | R)$, $\text{PMI}(n_1, a_1|R)$, $\text{PMI}(n_2, a_2|R)$, $\text{PMI}(\langle a_1, a_2 \rangle | R)$, $\text{PMI}(\langle a_1, a_2 \rangle, t|R)$, $\text{PMI}(a_1, t|R)$, $\text{PMI}(a_2, t|R)$, and again we obtain five features for p

from each one.

3.3.3.4 Token-based features

Another set of features operate on seed names, NP anchors, templates, and OWL identifiers of relations, viewed as sequences of tokens.

For each seed name n_1 and NP anchor a_1 that matched (as first elements of a seed name pair $\langle n_1, n_{i,2} \rangle$ and NP anchor pair $\langle a_1, a_{j,2} \rangle$) to produce a particular sentence plan p, we calculate their *cosine similarity* $\cos(n_1, a_1)$ with $tf \cdot idf$ weights (defined as in Section 3.3.1).[20] We then use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these cosine similarities as features of p. Intuitively, they show how good the matches that produced p were. We repeat for each seed name n_2 and NP anchor a_2 that matched (as second elements of their pairs) to produce p, this time computing $\cos(n_2, a_2)$, obtaining five additional features of p.

We do the same using the function $\text{AVGTOKPMI}(n_1, a_1)$ instead of $\cos(n_1, a_1)$, defining $\text{AVGTOKPMI}(n_1, a_1)$ as:

$$\text{AVGTOKPMI}(n_1, a_1 | R) = \frac{1}{|n_1| \cdot |a_1|} \sum_{\tau \in toks(n_1)} \sum_{\tau' \in toks(a_1)} \frac{1}{-\log P(\tau, \tau' | R)} \cdot \log \frac{P(\tau, \tau' | R)}{P(\tau | R) \cdot P(\tau' | R)} \tag{3.13}$$

where: $|n_1|$, $|a_1|$ are the lengths (in tokens) of n_1, a_1; $toks(n_1)$, $toks(a_1)$ are the token sequences of n_1, a_1, respectively; $P(\tau | R)$ is the probability of encountering τ in a parsed sentence of R; and $P(\tau, \tau' | R)$ is the probability of encountering both τ and τ' in the same parsed sentence of R; we use Laplace estimates for these probabilities. Again, we compute $\text{AVGTOKPMI}(n_1, a_1)$ for every seed name n_1 and NP anchor a_1 that matched to produce a particular sentence plan p, and we use the *maximum*, *minimum*,

[20] When computing the features of this section, all tokens are stemmed, stop-words are ignored, accents are removed, and numeric tokens are replaced by a particular pseudo-token, as in Section 3.3.1.

average, *total*, and *standard deviation* of these scores as features of p. We repeat using AVGTOKPMI(n_2, a_2) instead of $\cos(n_2, a_2)$, obtaining five more features.

In a similar manner, we compute AVGTOKPMI$(a_1, t|R)$ and AVGTOKPMI$(a_2, t|R)$ for each NP anchor a_1 or a_2 (left, or right element of an NP anchor pair) and template t (ignoring the S and O) that led to a particular sentence plan p, and we use their *maximum*, *minimum*, *average*, *total*, and *standard deviation* as ten additional features of p. We also compute $\cos(t, r)$ and AVGTOKPMI$(t, r|R)$ for each template t and tokenized identifier r of a relation R (e.g., $R =$ `madeFrom` becomes $r =$ "made from") that led to the sentence plan p , obtaining ten more features. Finally, we compute AVGTOKPMI$(a_1, a_2|R)$, AVGTOKPMI$(a_1, a_1|R)$, and AVGTOKPMI$(a_2, a_2|R)$ for all the a_1 and a_2 NP anchors (first or second elements in their pairs) of p, obtaining fifteen more features of p. Although they may look strange, in effect AVGTOKPMI$(a_1, a_1|R)$ and AVGTOKPMI$(a_2, a_2|R)$ examine how strongly connected the words inside each NP anchor (a_1 or a_2) are.

3.3.3.5 Other features

Another group of features try to estimate the grammaticality of a candidate sentence plan p. Let us assume that p is intended to express the relation R. For every seed name pair of R (not only seed name pairs that led to p), we generate a sentence using p; we ignore only seed name pairs that produced no sentence plans at all, which are assumed to be poor. For example, for the seed name pair $<n_1 =$"Semillon", $n_2 =$ "Semillon grape"$>$ of the relation $R =$ `madeFrom` and the following candidate sentence plan:

$$[ref(S)]^1_{nom}\ [\text{make}]^2_{verb,passive,present,agr=1,polarity=+}\ [\text{from}]^3_{prep}\ [ref(O)]^4_{acc}$$

the sentence "Semillon is made from Semillon grapes" is generated. We do not generate referring expressions, even when required by the sentence plan (e.g., $[ref(S)]^1_{nom}$); we use the seed names instead. We obtain confidence scores for these sentences from

the parser, and we normalize them dividing by each sentence's length. The *maximum*, *minimum*, *average*, and *standard deviation* of these scores are used as features of p.

Some additional features for a candidate sentence plan p follow:[21]

- True if p contains a present participle without an auxiliary; otherwise false. Sentence plans of this kind are often poor.
- True if p has a main verb in active voice; otherwise false.
- True if p contains a referring expression for S before a referring expression for O; otherwise false. Sentence plans that refer to S before O are usually simpler and better.
- True if a referring expression of p is the *subject* of a verb of p; otherwise false. This information is obtained from the parsed sentences that led to p. We use the most frequent dependency tree, if p was derived from many sentences. Sentence plans with no subjects are often ill-formed.
- True if a referring expression of p is the *object* of a verb of p; otherwise false. Again, this information is obtained from the parsed sentences that led to p, and we use the most frequent dependency tree. Sentence plans with no objects are often ill-formed, because most relations are expressed by transitive verbs.
- True if all the sentences that p was derived from were well-formed full sentences, according to the parser.
- True if p required repair at the end of the sentence plan generation (Section 3.3.2); otherwise false. Repaired sentence plans can be poor.
- The number of slots of p, the number of slots before the slot for S, the number of slots after the slot for O, the number of slots between the slots for S and O (4

[21] All the non-Boolean features that we use are normalized to $[0, 1]$.

features).

- The *maximum*, *minimum*, *average*, *total*, and *standard deviation* of the ranks of the Web pages (as returned by the search engine, Section 3.3.1) that contained the sentences p was obtained from. Sentences from higher-ranked Web pages are usually more relevant to the seed name pairs we use as queries. Hence, sentence plans obtained from higher-ranked Web pages are usually better.

- The number of Web pages that contained the sentences from which p was obtained. Uncommon sentences often lead to poor sentence plans.

3.3.4 Ranking the candidate sentence plans

Each candidate sentence plan of a relation R is represented as a feature vector $\vec{v}$, containing the 251 features discussed above. Each vector is given to the MaxEnt classifier to obtain a probability estimate $P(c_+|\vec{v},R)$ that it belongs in the positive class c_+, i.e., that the sentence plan is correct for R. The candidate sentence plans of each relation R are then ranked by decreasing estimated (by the classifier) $P(c_+|\vec{v},R)$. We call SP our overall sentence plan generation method that uses the probability estimates of the classifier to rank the candidate sentence plans.

In an alternative configuration of our sentence plan generation method, denoted SP*, the probability estimate $P(c_+|\vec{v},R)$ of each candidate sentence plan is multiplied by its *coverage*. To compute the coverage of a sentence plan for a relation R, we use the sentence plan to produce a sentence for each seed name pair of R (as when computing the grammaticality of a sentence plan in Section 3.3.3). Subsequently, we use each sentence as a phrase query in a Web search engine. The coverage of the sentence plan is the number of seed name pairs for which the search engine retrieved at least one document containing the search sentence (verbatim), divided by the total number of seed name pairs of R. Coverage is intended to help avoid sentence plans that produce

very uncommon sentences, usually an indication of incorrect sentence plans. Computing the coverage of *every* candidate sentence plan is time consuming, because of the Web searches; this is also why we do not include coverage in the features of the classifier. Hence, we first rank the candidate sentence plans of each relation R by decreasing $P(c_+|\vec{v},R)$, and we then re-rank only the top ten of them (per R) after multiplying the $P(c_+|\vec{v},R)$ of each one by its coverage.

In both SP and SP*, in a semi-automatic scenario we return to a human inspector the top five candidate sentence plans of each relation. In a fully automatic scenario, we return only the top one.

3.4 Experiments

We now present the experiments we performed to evaluate our methods that generate NL names and sentence plans.

3.4.1 The ontologies of our experiments

We used three ontologies for our experiments: (i) the Wine Ontology (Section 2.4.1), (ii) the M-PIRO Ontology, and (iii) the Disease Ontology (Section 2.4.3).

The M-PIRO Ontology, which we did not use in previous experiments, was originally developed in the M-PIRO project (Isard et al., 2003) and was later ported to OWL. It has been used to demonstrate the high quality texts that NaturalOWL can produce, when appropriate manually authored linguistic resources are provided (Galanis et al., 2009). We wanted to investigate if texts of similar quality can be generated with automatically or semi-automatically acquired NL names and sentence plans.

The current version of the M-PIRO Ontology contains 76 classes, 508 individuals, and 41 relations. Many individuals, however, are used to represent canned texts (e.g., manually written descriptions of particular types of exhibits) that are difficult

to generate from symbolic information. For example, there is a pseudo-individual `aryballos-def` whose NL name is the fixed string "An aryballos was a small spherical vase with a narrow neck, in which the athletes kept the oil they spread their bodies with". Several relations are also used only to link these pseudo-individuals (in effect, the canned texts) to other individuals or classes they describe (e.g., to link `aryballos-def` to the class `Aryballos`); and many other classes are used only to group pseudo-individuals (e.g., pseudo-individuals whose canned texts describe types of vessels all belong in a common class). In our experiments, we ignored pseudo-individuals, relations, and classes that are used to represent, link, and group canned texts, since we focus on generating texts from symbolic information. In our experiments, we aimed to produce NL names and sentence plans for the remaining 30 classes, 127 individuals, and 12 relations, which are all involved in the definitions (descriptions) of the 49 exhibits of the collection the ontology is about.

For the Wine Ontology, we aimed to produce NL names and sentence plans for the 49 classes, 146 individuals, and 7 relations that are directly involved in non-trivial definitions of wines (43 definitions of wine classes, 52 definitions of wine individuals). By "non-trivial definitions" we mean that we ignored wine definitions that humans understand as repeating information that is obvious from the name of the defined class or individual (e.g., the definition of the class `RedWine` in effect says that a red wine is a wine with red color). For the Disease Ontology, we aimed to automatically produce NL names and sentence plans for the 94 classes, 99 individuals, and 8 relations of the ontology that are involved in the definitions of 30 randomly selected diseases.

Manually authored NL names and sentence plans for the same classes, individuals, and relations, to be able to compare the quality of the resulting texts, are available from previous experiments (see Sections 2.4.1 and 2.4.2); specifically for the M-PIRO Ontology, resources were available from previous work (Galanis et al., 2009) and were re-authored for NaturalOWL version 2.

We note that datatype relations, where O is a datatype value (e.g., integer, string, date), can in principle be handled using the same methods, but appropriate recognizers may be needed to obtain appropriate anchors, instead of NP anchors. For example, a datatype relation may map persons to dates of birth; then a recognizer of dates would be needed to extract (and possibly normalize) appropriate date anchors from Web pages, since a parser may not treat dates as NPs.

3.4.2 Experiments with automatically or semi-automatically produced NL names

We now present the experiments we performed to evaluate our method that generates NL names (Section 3.2).

3.4.2.1 Anonymity experiments

Recall that when defining NL names, NaturalOWL allows declaring that an individual or class should be *anonymous*, i.e., that NaturalOWL should avoid referring to it by name (see Section 2.3.2.3). Our method that automatically produces NL names can also declare particular individuals or classes as anonymous (Section 3.2.1.1).

In a first experiment, we measured how well our NL names method determines which individuals and classes should be anonymous. We compared the decisions of our method against the corresponding anonymity declarations in the manually authored NL names of the three ontologies. Table 3.1 summarizes the results of this experiment. *Precision* is the total number of individuals and classes our NL names method *correctly* (in agreement with the manually authored NL names) declared as anonymous, divided by the total number of individuals and classes our method declared as anonymous. *Recall* is the total number of individuals and classes our NL names method correctly declared as anonymous, divided by the total number of individuals and classes (among

	WINE	M-PIRO	DISEASE
precision	1.00 (38/38)	1.00 (49/49)	undefined (0/0)
recall	0.73 (38/52)	1.00 (49/49)	undefined (0/0)
accuracy	0.93 (184/198)	1.00 (157/157)	1.00 (195/195)

Table 3.1: Results of the anonymity experiments.

those we aimed to produce NL names for) that the manually authored NL names declared as anonymous. For the Disease Ontology, the manually authored NL names and our NL names method agreed that no individuals and classes (that we aimed to produce NL names for) should be anonymous, which is why precision and recall are undefined. *Accuracy* is the number of correct decisions (individuals and classes correctly declared, or correctly not declared as anonymous), divided by the total number of individuals and classes (that we aimed to produce NL names for).

The anonymity decisions of our method were perfect in the M-PIRO Ontology and Disease Ontology. In the Wine Ontology, the precision of our method was also perfect, i.e., whenever our method decided to declare an individual or class as anonymous, this was a correct decision; but recall was lower, i.e., our method did not anonymize all the individual and classes that the manually authored NL names did. The latter is due to the fact that the manually authored NL names of the Wine Ontology also anonymize 14 individuals and classes with complex identifiers to produce more readable texts (e.g., `SchlossVolradTrochenbierenausleseRiesling`). By contrast, our method declares individuals and classes as anonymous only to avoid redundancy in the generated texts (Section 3.2.1.1), hence it does not anonymize the 14 individuals and classes.

3.4.2.2 Inspecting the produced NL names

We then invoked our NL name generation method for the individuals and classes it had not declared as anonymous (160 in the Wine Ontology, 108 in the M-PIRO Ontology, 195 in the Disease Ontology), using the top 10 returned documents per Web search (or

top 20, when the search engine proposed spelling corrections) (Section 3.2.1.3). We ranked the produced NL names (as in Sections 3.2.1.3 and 3.2.2), and kept the top 5 NL names per individual or class. The author of this thesis then inspected the resulting NL names and marked each one as correct or incorrect. An NL name was considered correct if and only if: (i) it would produce morphologically, syntactically, and semantically correct and unambiguous noun phrases for the corresponding individual or class (e.g., "Cabernet Sauvignon grape" is correct for `CabernetSauvignonGrape`, but "Cabernet Sauvignon wine", "Cabernet Sauvignon", or "grape" are incorrect); and (ii) the annotations of the slots of the NL name (e.g., POS tags, numbers, genders, agreement) were all correct.

Table 3.2 shows the results of this experiment. The "*1-in-1*" score is the ratio of individuals and classes for which the top returned NL name was correct. The "*1-in-3*" score is the ratio of individuals and classes for which there was at least one correct NL name among the top three, and similarly for "*1-in-5*". The "1-in-1" score corresponds to a fully automatic scenario, where the top NL name would be used for each individual or class, without human intervention. By contrast, the "1-in-3" and "1-in-5" scores correspond to a semi-automatic scenario, where a human would inspect the top three or five, respectively, NL names per individual or class, looking for a correct one to select. The *mean reciprocal rank* (MRR) is the mean (over all the individuals and classes we asked our method to produce NL names for) of the reciprocal rank $r_i = 1/k_i$, where k_i is the rank (1 to 5) of the top-most correct NL name returned for the i-th individual or class; if no correct NL name exists among the top five, then $r_i = 0$.We report MRR scores mostly to make it easier to compare future NL name methods against ours; MRR rewards more those methods that place correct NL names towards the top of the five returned NL names. The *weighted* scores of Table 3.2 are similar, but they weigh (when computing the ratios or mean) each individual or class by the number of OWL statements that mention it in the ontology, thus rewarding more heavily correct NL names of individuals

	WINE	M-PIRO	DISEASE
1-in-1	0.69 (110/160)	0.77 (83/108)	0.74 (145/195)
1-in-3	0.93 (145/160)	0.94 (102/108)	0.89 (173/195)
1-in-5	0.95 (152/160)	0.95 (103/108)	0.90 (176/195)
MRR	0.79	0.85	0.80
weighted 1-in-1	0.69	0.77	0.76
weighted 1-in-3	0.93	0.94	0.91
weighted 1-in-5	0.96	0.95	0.93
weighted MRR	0.80	0.85	0.82

Table 3.2: Results of the inspection of the produced NL names.

and classes that are more frequently used in the ontology.

Overall, the results of Table 3.2 indicate that our NL names method performs very well in a semi-automatic scenario where a human inspects and selects among the top five, or even only the top three, produced NL names per individual or class. In a fully automatic scenario, however, where the top NL name is used without human inspection, there is still large scope for improvement. We note, however, that our definition of correctness (of NL names) in the experiment of this section was particularly strict. For example, an NL name with only a single error in its slot annotations (e.g., a wrong gender in a noun slot) was counted as incorrect, even if in practice the error might have a minimal effect on the generated texts that would use the NL name. The experiment of Section 3.4.2.4 below, where NL names are considered in the context of generated texts, sheds more light on this point.

By inspecting the produced NL names, we noticed that our method is very resilient to spelling errors and abbreviations in the OWL identifiers of individuals and classes. For example, it returns NL names producing "a Côte d'Or wine" for `CotesDOr`, and "the Naples National Archaeological Museum" for `national-arch-napoli`. Several wrongly produced NL names are due to errors of the tools that our method in-

vokes (e.g., parser, named entity recognizer). Several other errors were due to over-shortened `altTokNames` (Section 3.2.1); for example, one of the `altTokNames` of `CorbansDryWhiteRiesling` was simply "Dry White", which leads to an NL name that does not identify the particular wine clearly enough. Another source of problems were OWL identifiers containing words with multiple possible POS tags (e.g., `gold`, where "gold" can be either a noun or adjective); in these cases, our method produces multiple candidate NL names with minor differences, one for each possible POS tag, which clutter the list of top five NL names, sometimes not allowing better NL names to surface. Finally, in the Disease Ontology, our automatic conversion produced many individuals whose identifiers are in effect long phrases (see, for example, the OWL description of `DOID_1328` on page 71). Our NL names method manages to produce appropriate NL names (with correct slot annotations etc.) for some of them (e.g., `mutation_in_the_SLC26A2_gene`), but produces no NL names in other cases (e.g., `infection_of_the_keratinized_layers`). Some of these errors, however, may not have a significant effect on the generated texts (e.g., using the tokenized identifer "infection of the keratinized layers", which is the default when no NL name is provided, may still lead to a reasonable text). Again, the experiment of Section 3.4.2.4 below sheds more light on this point.

3.4.2.3 Effort to semi-automatically author NL names and inter-annotator agreement

The top five automatically produced NL names of each individual and class were also shown to a second human judge. The second judge was a computer science researcher not involved in NLG, fluent in English, though not a native speaker. For each individual or class, and for each one of its top five NL names, the judge was shown a phrase produced by the NL name (e.g., "Cabernet Sauvignon"), an automatically generated sentence about the individual or class (expressing a message triple of the ontology)

illustrating the use of the NL name (e.g., "Cabernet Sauvignon is a kind of wine."), and an additional sentence where the NL name had been automatically replaced by a pronoun (e.g., "It is a kind of wine.") to check the gender of the NL name. The judge was asked to consider the phrases and sentences, and mark the best correct NL name for each individual or class. The judge could also mark more than one NL names for the same individual or class, if more than one seemed correct and equally good; the judge was instructed not to mark any of the five NL names, if none seemed correct. The judge completed this task in 49, 45, and 75 minutes for the Wine (727 candidate NL names), M-PIRO (540 candidate NL names), and Disease Ontology (965 candidate NL names), respectively; by contrast, manually authoring the NL names of the three ontologies took approximately 2, 2, and 3 working days, respectively. These times and the fact that the second judge was not acquainted with the inner workings of the NLG system, nor with the details of its linguistic resources, suggest that the semi-automatic authoring scenario is viable and very useful in practice.

Table 3.3 compares the decisions of the second judge, hereafter called J_2, to those of the author of this thesis, hereafter called J_1. J_1 was able to view the full details of the NL names using NaturalOWL's Protégé plug-in, unlike J_2 who viewed only phrases and example sentences. For the purposes of this study, J_1 marked *all* the correct NL names (not only the best ones) among the top five of each individual or class. In Table 3.3, *micro-precision* is the number of NL names (across all the individuals and classes) that were marked as correct by both J_1 and J_2, divided by the number of NL names marked as correct by J_2, i.e., we treat the decisions of J_1 as gold. *Macro-precision* is similar, but we first compute the precision of J_2 against J_1 separately for each individual or class, and we then average over all the individuals and classes. J_1 *1-in-5* is the percentage of individuals and classes for which J_1 marked at least one NL name among the top five as correct. J_2 *1-in-5* is the percentage of individuals and classes for which J_2 marked at least one NL name among the top five as correct. *Pseudo-recall* is the number of

	WINE	M-PIRO	DISEASE
micro-precision	0.97	0.96	0.97
macro-precision	0.98	0.94	0.98
J_1 1-in-5	0.95	0.95	0.90
J_2 1-in-5	0.95	0.93	0.90
pseudo-recall	1.00	0.96	1.00
Cohen's Kappa	0.77	0.80	0.98

Table 3.3: Inter-annotator agreement in the semi-automatic authoring of NL names.

individuals and classes for which both J_1 and J_2 marked at least one NL name as correct, divided by the number of individuals and classes for which J_1 marked at least one NL name as correct; this measure shows how frequently J_2 managed to find at least one correct (according to J_1) NL name, when there was at least one correct NL name among the top five. Computing the true recall of the decisions of J_2 against those of J_1 would be inappropriate, because J_2 was instructed to mark only the best NL name(s) of each individual and class, unlike J_1 who was instructed to mark all the correct ones. We also calculated Cohen's Kappa between J_1 and J_2; for each individual or class, if J_1 had marked more than one NL names as correct, we kept only the top-most, and similarly for J_2, hence each judge had six possible choices (including marking no NL name) per individual and class. The results of Table 3.3 indicate strong inter-annotator agreement in the semi-automatic authoring of NL names in all three ontologies.

3.4.2.4 Evaluating NL names in generated texts

In order to examine how the produced NL names affect the perceived quality of the generated texts, we showed automatically generated texts describing individuals and classes of the three ontologies to six computer science students not involved in the work of this article; they were all fluent, though not native, English speakers. We generated texts using NaturalOWL configured in four ways. The NO-NLN configuration uses no

NL names; in this case, NaturalOWL uses the tokenized OWL identifiers of the individuals and classes as their names.[22] MANUAL-NLN uses manually authored NL names. AUTO-NLN uses the top-ranked NL name that our NL names method produces for each individual and class. Finally, SEMI-AUTO-NLN uses the NL name (of each individual or class) that a human inspector (the author of this thesis) selected among the top five NL names produced by our method. Additionally, both AUTO-NLN and SEMI-AUTO-NLN use the methods of Sections 3.2.1.1 and 3.2.3 to anonymize individuals or classes and to infer interest scores from NL names, whereas MANUAL-NLN uses the anonymity declarations and interest scores of the manually authored linguistic resources, and NO-NLN uses no anonymity declarations and no interest scores. Apart from the NL names, anonymity declarations, and interest scores, all four configurations use the same, manually authored other types of linguistic resources (e.g., sentence plans, text plans to order the message triples). Below are example texts generated from the three ontologies by the four configurations.

MANUAL-NLN: This is a moderate, dry Zinfandel. It has a medium body. It is made by Saucelito Canyon in the city of Arroyo Grande.

SEMI-AUTO-NLN: This is a moderate, dry Zinfandel wine. It has a medium body. It is made by the Saucelito Canyon Winery in the Arroyo Grande area.

AUTO-NLN: This is a dry Zinfandel and has the medium body. It is the moderate. It is made by Saucelito Canyon in Arroyo Grande.

NO-NLN: Saucelito Canyon Zinfandel is Zinfandel. It is Dry. It has a Medium body. It is Moderate. It is made by Saucelito Canyon. It is made in Arroyo Grande Region.

MANUAL-NLN: This is a statue, created during the classical period and sculpted by Polykleitos. Currently it is exhibited in the National Archaeological Museum of Napoli.

SEMI-AUTO-NLN: This is a statue, created during the classical period and sculpted by the sculp-

[22] If the ontology provides an `rdfs:label` for an individual or class, NO-NLN uses a tokenized form of the `rdfs:label`.

tor polyclitus. Currently it is exhibited in the Naples National Archaeological Museum.

AUTO-NLN: This is a statue, created during classical and sculpted by the polyclitus. Currently it is exhibited in national arch napoli.

NO-NLN: Exhibit 4 is statue, created during classical period and sculpted by polyclitus. Today it is exhibited in national arch napoli.

MANUAL-NLN: Systemic mycosis is a kind of fungal infectious disease that affects the human body. It results in infection of internal organs. It is caused by fungi.

SEMI-AUTO-NLN: A systemic mycosis is a kind of fungal infectious disease that affects human body. It results in infections of internal organs and it is caused by the fungi.

AUTO-NLN: A systemic mycosis is fungal that affects human body. It results in infections of internal organs and it is caused by the fungi.

NO-NLN: Systemic mycosis is a kind of fungal infectious disease. It affects human body. It results in infection of internal organs. It is caused by Fungi.

We note that some NL names of SEMI-AUTO-NLN and AUTO-NLN can be easily improved using the Protégé plug-in of NaturalOWL. For example, the NL name of the human body can be easily modified to include a definite article, which would improve the texts of SEMI-AUTO-NLN and AUTO-NLN in the Disease Ontology examples above ("affects the human body" instead of "affects human body").[23] Nevertheless, we made no such improvements.

Recall that there are $43 + 52 = 95$ non-trivial definitions of wine classes and wine individuals in the Wine Ontology, 49 exhibits in the M-PIRO Ontology, and that we randomly selected 30 diseases from the Disease Ontology (Section 3.4.1). Hence, we generated $95 \times 4 = 380$ texts from the Wine Ontology (with the four configurations of NaturalOWL), $49 \times 4 = 196$ texts from the M-PIRO Ontology, and $30 \times 4 = 120$ texts from the Disease ontology. For each individual or class that we generated texts for,

[23]The missing article is due to the fact that the on-line dictionary we used marks "body" as (possibly) non-countable.

the message triples of its definition (regardless of their interest scores) along with the corresponding texts produced by the four configurations of NaturalOWL were given to exactly one student. The four texts of each individual or class were randomly ordered and the students did not know which configuration had generated each one of the four texts. For each individual or class, the students were asked to compare the four texts to each other and to the message triples, and score each text by stating how strongly they agreed or disagreed with statements S_1–S_4 below. A scale from 1 to 5 was used (1: strong disagreement, 3: ambivalent, 5: strong agreement). Examples and more detailed guidelines were also provided to the students.

(S_1) *Sentence fluency*: Each sentence of the text (on its own) is grammatical and sounds natural.

(S_2) *Clarity*: The text is easy to understand, provided that the reader is familiar with the terminology and concepts of the domain (e.g., historical periods, grape varieties, virus names).

(S_3) *Semantic correctness*: The text accurately conveys the information of the message triples.

(S_4) *Non-redundancy*: There is no redundancy in the text (e.g., stating the obvious or repeating information).

Tables 3.4–3.6 show the scores of the four configurations of NaturalOWL, averaged over the texts of each ontology. For each criterion, the best scores are shown in bold. We performed Analysis of Variance (ANOVA) and post-hoc Tukey tests to check for statistically significant differences ($a = 0.05$) in the four scores of each criterion in each ontology. In each criterion (row), we detected no statistically significant differences between scores marked with the same superscript; all the other differences (in the same row) were statistically significant. A post-hoc power analysis of the ANOVA values resulted in power values greater or equal to 0.98, with the exception of the non-redundancy scores of the Disease Ontology (last row of Table 3.6), where power was

0.25.

Overall, the manually authored NL names led to the best (near-perfect) scores, as one might expect. The scores of SEMI-AUTO-NLN were overall slightly lower, but still high (always greater or equal to $4.3/5$) and no statistically significant differences to the corresponding scores of MANUAL-NLN were detected. These findings confirm the conclusion of Section 3.4.2.2 that our NL names method performs very well in a semi-automatic scenario, where a human inspects and selects among the top-ranked automatically produced NL names. By contrast, AUTO-NLN performed overall much worse than SEMI-AUTO-NLN and MANUAL-NLN, and often worse than NO-NLN, which confirms the conclusion of Section 3.4.2.2 that our NL names method cannot be used in a fully automatic manner.

The NO-NLN configuration, which uses tokenized identifiers of individuals and classes, performed overall much worse than MANUAL-NLN and SEMI-AUTO-NLN in the Wine and M-PIRO ontologies, which shows the importance of NL names in the perceived quality of generated texts. The differences between NO-NLN, SEMI-AUTO-NLN, and MANUAL-NLN were smaller in the Disease Ontology, where no statistically significant differences between the three configurations were detected. These smaller differences are explained by the fact that the conversion of the Disease Ontology (Section 2.4.3) produced many individuals whose OWL identifiers are in effect long phrases (e.g., `infection_of_the_keratinized_layers`), easily readable, and sometimes better than then top-ranked NL names; furthermore, our NL names method does not manage to produce any NL names for many of these individuals and, hence, SEMI-AUTO-NLN ends up using their tokenized identifiers, like NO-NLN. We also note that there are very few redundant message triples and no anonymous individuals or classes in the Disease Ontology, which explains the higher non-redundancy scores of NO-NLN in the Disease Ontology, compared to the much lower non-redundancy scores of NO-NLN in the other two ontologies.

Criteria	NO-NLN	AUTO-NLN	SEMI-AUTO-NLN	MANUAL-NLN
Sentence fluency	3.99	3.50	**4.70**[1]	**4.83**[1]
Clarity	4.41	3.43	**4.79**[1]	**4.79**[1]
Semantic correctness	4.44[1]	3.54	**4.66**[1,2]	**4.85**[2]
Non-redundancy	3.17	3.93[1]	**4.31**[1,2]	**4.56**[2]

Table 3.4: Human scores for texts generated from the Wine Ontology with different methods to obtain NL names.

Criteria	NO-NLN	AUTO-NLN	SEMI-AUTO-NLN	MANUAL-NLN
Sentence fluency	4.14[1]	4.22[1]	**4.90**[2]	**4.98**[2]
Clarity	4.00[1]	3.69[1]	**4.82**[2]	**4.98**[2]
Semantic correctness	4.06[1]	4.04[1]	**4.82**[2]	**4.98**[2]
Non-redundancy	3.18	4.06	**4.86**[1]	**4.96**[1]

Table 3.5: Human scores for texts generated from the M-PIRO Ontology with different methods to obtain NL names.

Criteria	NO-NLN	AUTO-NLN	SEMI-AUTO-NLN	MANUAL-NLN
Sentence fluency	**4.27**[1,2]	4.03[1]	**4.40**[1,2]	**4.73**[2]
Clarity	**4.73**[1]	3.80	**4.57**[1]	**4.90**[1]
Semantic correctness	**4.83**[1]	4.23[2]	**4.47**[1,2]	**4.87**[1]
Non-redundancy	**4.23**[1]	**4.30**[1]	**4.43**[1]	**4.33**[1]

Table 3.6: Human scores for texts generated from the Disease Ontology with different methods to obtain NL names.

3.4.3 Experiments with automatically or semi-automatically produced sentence plans

We now present the experiments we performed to evaluate our method that generates sentence plans (Section 3.3). Recall that our method employs a MaxEnt classifier to predict the probability that a candidate sentence plan is correct (positive class) or incorrect (negative class).[24]

3.4.3.1 Training the classifier of the sentence plan generation method

To create training instances for the MaxEnt classifier, we used our sentence plan generation method without the classifier to obtain candidate sentence plans (as in Sections 3.3.1 and 3.3.2) from Wikipedia for the seven relations of the Wine Ontology (Section 2.4.1). We used the manually authored NL names of the Wine Ontology to obtain seed names, and the top 50 Wikipedia articles of each search query.[25] We searched Wikipedia exclusively at this stage, as opposed to querying the entire Web, to obtain high quality texts and, hence, hopefully more positive training examples (correct sentence plans). The author of this thesis then manually tagged the resulting 655 candidate sentence plans as positive or negative training instances, depending on whether or not they were correct for the corresponding relation. A candidate sentence plan was considered correct if and only if: (i) it would produce morphologically, syntactically, and semantically correct sentences; and (ii) the annotations of its slots (e.g., POS tags, voice, tense, agreement) were all correct. To compensate for class imbalance in the training set (16% positive vs. 84% negative candidate sentence plans), we replicated all the positive training instances (over-sampling) to obtain an equal number of positive and negative

[24] We also experimented with SVMs and different kernels, but we saw no significant improvements compared to MaxEnt.

[25] We used Google's Custom Search API (`developers.google.com/custom-search/`) to search Wikipedia.

training instances.

Figure 3.3 shows the error rate of the classifier on (i) unseen instances (test error) and (ii) on the instances it has been trained on (training error). To obtain the curves of Fig. 3.3, we performed a leave-one-out cross validation on the 655 instances (candidate sentence plans) we had constructed, i.e., we repeated the experiment 655 times, each time using a different instance as the only test instance and the other 654 instances as the training dataset. Within each repetition of the cross-validation, we iteratively trained the classifier on 10%, 20%, ..., 100% of the training dataset (654 instances, with over-sampling applied to them). The *training error* counts how many of the instances that were used to train the classifier were also correctly classified by the classifier (trained on the same instances). The *test error* counts how many of the test (unseen) instances (one in each repetition of the cross-validation) were correctly classified by the classifier (trained on the corresponding percentage of the training dataset). The error rates of Fig. 3.3 are averaged over the 655 repetitions of the cross-validation.[26] The training error curve can be thought of as a lower bound of the test error curve, since a classifier typically performs better on the instances it has been trained than on unseen instances. The two curves indicate that the classifier might perform slightly better with more training data, though the test error rate would remain above 0.1. The relatively small distance of the right ends of the two curves indicates only mild overfitting when the entire training dataset is used.

To assess the contribution of the 251 features (Section 3.3.3), we ranked them by decreasing Information Gain (IG) (Manning and Schutze, 2000) computed on the 655 instances. Table 3.7 shows the maximum, minimum, and average IG scores of the features in each group (subsection) of Section 3.3.3. On average, the PMI and token-based features are the best predictors, whereas the prominence features are the worst.

[26] We set the similarity threshold T (Section 3.3.1) to 0.1, based on additional cross-validation experiments with different T values.

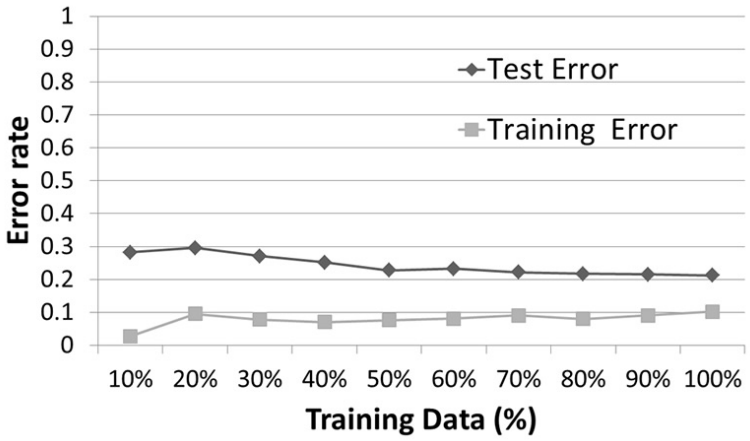


Figure 3.3: Test and training error rate of the classifier of our sentence plan generation method, for different sizes of training dataset.

The maximum scores, however, show that there is at least one good feature in each group, with the prominence features being again the worst in this respect. The minimum scores indicate that there is also at least one weak feature in every group, with the exception of the PMI features, where the minimum IG score (0.21) was much higher. Figure 3.4 shows the IG scores of all the features in each group, in ascending order. There are clearly many good features in every group, even in the 'other' group, which has the largest proportion of weak features.

We then iteratively trained the classifier on the entire training dataset (100%), removing at each iteration the feature (among the remaining ones) with the smallest IG score. Figure 3.5 shows the resulting test and training error rate curves, again obtained using a leave-one-out cross-validation. As more features are removed, the distance between the training and test error decreases, because of reduced overfitting. When very few features are left (far right), the performance of the classifier on unseen instances becomes unstable. The best results are obtained using all (or almost all) of the features, but the test error is almost stable from approximately 50 to 200 removed features, indicating that there is a lot of redundancy (e.g., correlated features) in the feature set.

	AVG INFORMATION GAIN	MAX INFORMATION GAIN	MIN INFORMATION GAIN
Productivity features	0.29	0.58	0.06
Prominence features	0.13	0.29	0.05
PMI features	0.50	0.62	0.21
Token-based features	0.48	0.61	0.03
Other features	0.20	0.62	0.00

Table 3.7: Information Gain of different groups of features used by the classifier of our sentence plan generation method.

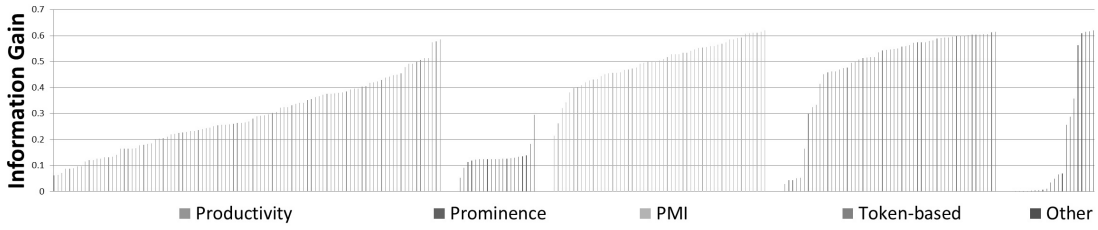


Figure 3.4: Ascending Information Gain scores in each group of features used by the classifier of our sentence plan generation method.

Nevertheless, we did not remove any features in the subsequent experiments, since the overfitting was reasonably low and the training and test times of the MaxEnt classifier were also low (performing a leave-one-out cross-validation on the 655 instances with all the features took approximately 6 minutes). We hope to explore dimensionality reduction further (e.g., via PCA) in future work.

3.4.3.2 Inspecting the produced sentence plans

In a subsequent experiment, the classifier was trained on the 655 instances (candidate sentence plans) of the previous section; recall that those instances were obtained from Wikipedia articles for Wine Ontology relations. The classifier was then embedded (without retraining) in our overall sentence plan generation method (SP or SP*, see Section 3.3.4). The sentence plan generation method (with the already trained classifier embedded) was then invoked to produce sentence plans from the entire Web, not just

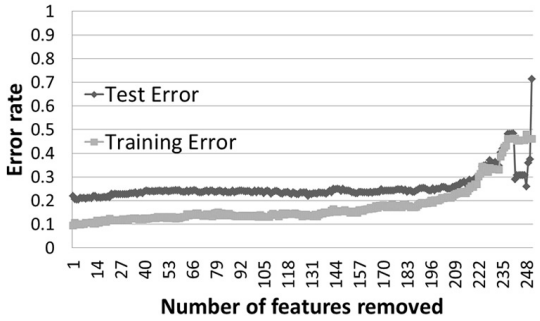


Figure 3.5: Test and training error rate of the classifier of our sentence plan generation method, for different numbers of features.

Wikipedia, and for the relations of all three ontologies, not just those of the Wine Ontology.[27] We kept the top 10 returned documents per Web search (Section 3.3.1) in the experiments of this section, to reduce the time needed to complete them. The author of this thesis inspected the resulting top five sentence plans of each relation (as ranked by SP or SP*), and marked them as correct or incorrect (as in Section 3.4.3.1). We then computed the *1-in1*, *1-in-5*, and MRR scores of the produced sentence plans per ontology, along with weighted variants of the three measures. All six measures are defined as in Section 3.4.2.2, but for sentence plans instead of NL names; the weighted variants weigh each relation by the number of OWL statements that mention it in the ontology.

Tables 3.8–3.10 show the results for the three ontologies. The configuration "SP with seeds of MANUAL-NLN" uses the NL names from the manually authored linguistic resources to obtain seed names (Section 3.3.1); "SP with seeds of SEMI-AUTO-NLN" uses the semi-automatically produced NL names to obtain seed names; "SP* with seeds of MANUAL-NLN" and "SP* with seeds of SEMI-AUTO-NLN" are similar, but they use SP* (which reranks the candidate sentence plans using their coverage) instead of SP.

[27] To avoid testing the classifier on sentence plans it had encountered during its training (more likely to happen with sentence plans for Wine Ontology relations), we actually excluded from the training data of the classifier any of the 655 instances that had the same feature vector with any test candidate sentence plan of the experiments of this section, when classifying that particular test candidate sentence plan.

	SP with seeds of MANUAL-NLN	SP* with seeds of MANUAL-NLN	SP with seeds of SEMI-AUTO-NLN	SP* with seeds of SEMI-AUTO-NLN	BOOT with seeds of MANUAL-NLN
1-in-1	0.71 (5/7)	**0.86** (6/7)	0.71 (5/7)	0.71 (5/7)	0.57 (4/7)
1-in-3	0.86 (6/7)	0.86 (6/7)	**1.00** (7/7)	**1.00** (7/7)	0.71 (5/7)
1-in-5	**1.00** (7/7)	**1.00** (7/7)	**1.00** (7/7)	**1.00** (7/7)	0.86 (6/7)
MRR	0.82	**0.89**	0.83	0.86	0.68
weighted 1-in-1	0.73	**0.86**	0.71	0.71	0.58
weighted 1-in-3	0.86	0.86	**1.00**	**1.00**	0.71
weighted 1-in-5	**1.00**	**1.00**	**1.00**	**1.00**	0.86
weighted MRR	0.83	**0.89**	0.83	0.86	0.68

Table 3.8: Results of the inspection of the produced sentence plans for the Wine Ontology.

Tables 3.8–3.10 also include results for a bootstrapping baseline (BOOT), described below. For each measure, the best results are shown in bold.

Overall, SP* performs better than SP, though the scores of the two methods are very close or identical in many cases, and occasionally SP performs better. Also, SP and SP* occasionally perform better when semi-automatically produced NL names are used to obtain seed names ("with seeds of SEMI-AUTO-NLN"), than when manually authored NL names are used ("with seeds of MANUAL-NLN"). It seems that manually authored NL names occasionally produce seeds that are uncommon on the Web and, hence, do not help produce good sentence plans, unlike semi-automatically produced NL names, which are extracted from the Web (and then manually filtered). The 1-in-5 results of Tables 3.8–3.10 clearly indicate that our sentence plan generation method (especially SP*) performs very well in a semi-automatic scenario, especially if the weighted measures are considered. By contrast, our method does not always perform well in a fully automatic scenario (1-in-1 results); the Disease Ontology was the most difficult in that respect. Overall, SP* with seeds of SEMI-AUTO-NLN seems to be the best version. The MRR scores of our method (all versions) were higher in the Wine Ontology and lower in the other two ones, which may be due to the fact that the classifier was trained for

	SP with seeds of MANUAL-NLN	SP* with seeds of MANUAL-NLN	SP with seeds of SEMI-AUTO-NLN	SP* with seeds of SEMI-AUTO-NLN	BOOT with seeds of MANUAL-NLN
1-in-1	0.58 (7/12)	**0.67** (8/12)	**0.67** (8/12)	**0.67** (8/12)	0.17 (2/12)
1-in-3	0.67 (8/12)	0.67 (8/12)	**0.75** (9/12)	**0.75** (9/12)	0.33 (4/12)
1-in-5	0.67 (8/12)	0.67 (8/12)	**0.83** (10/12)	**0.83** (10/12)	0.33 (4/12)
MRR	0.62	0.67	**0.73**	**0.73**	0.24
weighted 1-in-1	0.73	**0.85**	0.61	0.72	0.04
weighted 1-in-3	**0.85**	**0.85**	0.73	0.73	0.19
weighted 1-in-5	0.85	0.85	**0.98**	**0.98**	0.45
weighted MRR	0.79	**0.85**	0.73	0.79	0.22

Table 3.9: Results of the inspection of the produced sentence plans for the M-PIRO Ontology.

	SP with seeds of MANUAL-NLN	SP* with seeds of MANUAL-NLN	SP with seeds of SEMI-AUTO-NLN	SP* with seeds of SEMI-AUTO-NLN	BOOT with seeds of MANUAL-NLN
1-in-1	0.00 (0/8)	0.37 (3/8)	0.12 (1/8)	**0.50** (4/8)	0.12 (1/8)
1-in-3	0.75 (6/8)	**0.87** (7/8)	0.75 (6/8)	**0.87** (7/8)	0.37 (3/8)
1-in-5	**1.00** (8/8)	**1.00** (8/8)	**1.00** (8/8)	**1.00** (8/8)	0.75 (6/8)
MRR	0.41	0.65	0.47	**0.70**	0.33
weighted 1-in-1	0.00	0.16	0.01	**0.17**	0.00
weighted 1-in-3	0.42	**0.89**	**0.89**	**0.89**	0.15
weighted 1-in-5	**1.00**	**1.00**	**1.00**	**1.00**	0.85
weighted MRR	0.35	**0.55**	0.45	0.48	0.22

Table 3.10: Results of the inspection of the produced sentence plans for the Disease Ontology.

Wine Ontology relations (but with texts from Wikipedia).

While inspecting the automatically generated sentence plans, we noticed that there were several cases where the OWL identifier of the relation was poor (e.g., the relation `locatedIn` of the Wine Ontology connects wines to the regions they are produced in), but our method was still able to produce good sentence plans (e.g., [*ref*(*S*)] [is produced] [in] [*ref*(*O*)]). On the other hand, our method (all versions) produced very few (or none) sentence plans for relations with fewer than 10 seed name pairs. Also, the most commonly produced sentence plans are [*ref*(*S*)] [is] [*ref*(*O*)] and [*ref*(*O*)] [is] [*ref*(*S*)]. While the former may be an appropriate sentence plan for a message triple $\langle S, R, O \rangle$, the latter is almost never appropriate, so we always discard it.

3.4.3.3 Effort to semi-automatically author sentence plans and inter-annotator agreement

The top five sentence plans of SP* for each relation were also shown to the second human judge (J_2) who had examined the automatically produced NL names in the experiments of Section 3.4.2.3. For each relation, and for each one of its top five sentence plans, the second judge was shown a template view of the sentence plan (e.g., "*S* is made from *O*"), and an automatically generated sentence illustrating the use of the sentence plan (e.g., "Cabernet Sauvignon is made from Cabernet Sauvignon grapes."). The judge was asked to consider the templates and sentences, and mark the best correct sentence plan for each relation. The judge could also mark more than one sentence plans for the same relation, if more than one seemed correct and equally good; the judge was instructed not to mark any of the five sentence plans, if none seemed correct. The second judge completed this task (inspecting 40, 29, and 40 candidate sentence plans of the Wine, M-PIRO, and Disease Ontology, respectively) in approximately 5 minutes per ontology (15 minutes for all three ontologies); by contrast, manually authoring the sentence plans took approximately one working day per ontology. Again, these times

	WINE	M-PIRO	DISEASE
micro-precision	1.00	1.00	1.00
macro-precision	1.00	1.00	1.00
J_1 1-in-5	1.00	0.83	1.00
J_2 1-in-5	1.00	0.75	1.00
pseudo-recall	1.00	0.75	1.00
Cohen's Kappa	1.00	0.62	0.74

Table 3.11: Inter-annotator agreement in the semi-automatic authoring of sentence plans.

suggest that the semi-automatic authoring scenario is viable and very useful in practice.

We also measured the agreement between the second judge (J_2) and the author of this thesis (J_1) in the semi-automatic authoring (selection) of sentence plans, as in Section 3.4.2.3. The results, reported in Table 3.11, show perfect agreement in the Wine and Disease Ontologies. In the M-PIRO ontology, the agreement was lower, but still reasonably high; the pseudo-recall score shows J_2 did not select any sentence plan for some relations where J_1 believed there was a correct one among the top five.

3.4.3.4 The sentence plan generation baseline that uses bootstrapping

As a baseline, we also implemented a sentence plan generation method that uses a bootstrapping template extraction approach. Bootstrapping is often used in information extraction to obtain templates that extract instances of a particular relation (e.g., `makeFrom`) from texts, starting from a number of seed pairs of entity names (e.g., <"cream", "milk">) for which the relation is known to hold (Riloff and Jones, 1999; Muslea, 1999; Xu et al., 2007; Gupta and Manning, 2014). The seed pairs are used as queries in a search engine to obtain documents that contain them in the same sentence (e.g., "cream is made from milk"). Templates are then obtained by replacing the seeds with slots in the retrieved sentences (e.g., "*X* is made from *Y*"). The templates (without

their slots, e.g., "is made from") are then used as phrasal search queries to obtain new sentences (e.g., "gasoline is made from petroleum"), from which new pairs of entity names (<"gasoline", "petroleum">) are obtained. A new iteration can then start with the new pairs as seeds, leading to new templates, and so on.

Given a relation R, our baseline, denoted BOOT, first constructs seed name pairs using the ontology and the NL names, as in Section 3.3.1; we used only manually authored NL names in the experiments with BOOT. Then, BOOT uses the seed name pairs to obtain templates ("X is made from Y") from the Web, again as in Section 3.3.1. If the number of obtained templates is smaller than L, the templates (without their slots, e.g., "is made from") are used as phrasal search queries to obtain new documents and new sentences (from the documents) that match the templates. For each new sentence (e.g., "gasoline is made from petroleum"), BOOT finds the NPs ("gasoline", "petroleum") immediately before and after the search phrase, and treats them as a new seed name pair discarding seed name pairs that occur in only one retrieved document. The new seed name pairs are then used to obtain new templates; again, templates that occur in only one retrieved document are discarded. This process is repeated until we have at least L templates for R, or until no new templates can be produced. In our experiments, we set $L = 150$ to obtain approximately the same number of templates as with SP and SP*.

At the end of the bootstrapping, instead of using a MaxEnt classifier (Sections 3.3.3 and 3.3.4), BOOT scores the templates of each relation R using the following confidence function:[28]

$$conf(t) = \frac{hits(t)}{hits(t) + misses(t)} \cdot \log finds(t) \tag{3.14}$$

[28]This function is from Chapter 22 of (Jurafsky and Martin, 2008), and is based on a similar function of (Riloff and Jones, 1999). Unlike Jurafsky and Martin, we count NP anchor pairs rather than seed name pairs in $hits(t)$ and $misses(t)$, which places more emphasis on extracting all the possible NPs (of the retrieved documents of R) that correspond to the seed names.

where t is a template being scored, $hits(t)$ is the number of (distinct) anchor pairs of R extracted by t (from the documents retrieved by all the seed name pairs of R), $misses(t)$ is the number of (distinct) anchor pairs of R *not* extracted by t (from the same documents), and $finds(t)$ is the number of sentences (of the same documents) that match t. The five templates with the highest $conf(t)$ (in a semi-automatic scenario) or the single template with the highest $conf(t)$ (in a fully automatic scenario) are then converted to sentence plans, as in Section 3.3.2.

Functions like Eq. 3.14 can also be applied *within* each iteration of the bootstrapping, not only at the end of the entire bootstrapping, to keep only the best new templates of each iteration. This may help avoid concept drift, i.e., gradually obtaining templates that are more appropriate for other relations that just happen to share several seed name pairs with the relation we wish to generate templates for. We did not use Eq. 3.14 within each iteration of the bootstrapping, because in our experiments very few iterations (most often only the initial one) were needed. Also, using a function like Eq. 3.14 within each iteration requires a threshold d, to discard templates with $conf(t) < d$ at the end of each iteration, which in practice is not trivial to tune. Similar functions can be used to score the new seed name pairs within each iteration or at the end of the bootstrapping.[29] Since very few iterations (most often only the initial one) were needed in our experiments, we ended up using mostly (and most often only) the initial seed name pairs, which are known to be correct; hence, scoring the seed name pairs seemed unnecessary.

Tables 3.8–3.10 show that the results of BOOT are consistently worse than the results of SP and SP*. As already noted, for most relations more than L templates had been produced at the end of the first iteration (with the initial seed name pairs) of BOOT. Additional iterations were used only for 5 relations of the M-PIRO Ontology. Hence,

[29] Consult Chapter 22 of (Jurafsky and Martin, 2008) for a broader discussion of bootstrapping template extraction methods. See also `http://nlp.stanford.edu/software/patternslearning.shtml` for a publicly available bootstrapping template extraction system (SPIED), which supports however only templates with a single slot (e.g., "X is a city"), unlike the templates we need.

the differences in the performance of BOOT compared to SP and SP* are almost entirely due to the fact that BOOT uses the confidence function of Eq. 3.14 instead of the MaxEnt classifier (and the coverage of the sentence plans, in the case of SP*). Hence, the MaxEnt classifier has an important contribution in the performance of SP and SP*. Tables 3.8–3.10 show that this contribution is large in all three ontologies, despite the fact that the classifier was trained on Wine Ontology relations only (but with texts from Wikipedia).

3.4.3.5 Evaluating sentence plans in generated texts

To examine how sentence plans produced by different methods affect the perceived quality of generated texts, we showed automatically generated texts describing individuals and classes of the three ontologies to six computer science students, the same students as in the experiments of Section 3.4.2.4. We used six configurations of NaturalOWL in this experiment. The NO-SP configuration is given no sentence plans; in this case, NaturalOWL automatically produces sentence plans by tokenizing the OWL identifiers of the relations, acting like a simple verbalizer.[30] MANUAL-SP uses manually authored sentence plans. AUTO-SP* uses the SP* method (Section 3.3.4) with no human selection of sentence plans, i.e., the top-ranked sentence plan of each relation. We did not consider SP in this experiment, since the experiments of the previous section indicated that SP* was overall better. In SEMI-AUTO-SP*, a human inspector (the author of this thesis) selected the best sentence plan of each relation among the five top-ranked sentence plans produced by SP*. Similarly, AUTO-BOOT and SEMI-AUTO-BOOT use the BOOT baseline of Section 3.4.3.4 with no human selection or with a human inspector selecting among the top five, respectively. Apart from the sentence plans, all six configurations use the same, manually authored other types of linguistic resources (e.g.,

[30] If the ontology provides an `rdfs:label` for a relation, NO-SP produces a sentence plan by tokenizing the `rdfs:label`.

NL names, interest scores, text plans to order the message triples). Below are example texts generated from the three ontologies by the six configurations.

MANUAL-SP: This is a moderate, dry Zinfandel. It has a full body. It is made by Elyse in the Napa County.

SEMI-AUTO-SP*: This is a full, dry Zinfandel. It is moderate. It is made at Elyse in the Napa County.

SEMI-AUTO-BOOT: This is a full, dry Zinfandel. It is moderate. Elyse produced it and the Napa County is Home to it.

AUTO-SP*: This is a full, dry Zinfandel. It is moderate. It is made at Elyse and it is the Napa County.

AUTO-BOOT: This is a full, dry Zinfandel. It is moderate. It is Elyse and the Napa County.

NO-SP: This is a Zinfandel. It has sugar dry, it has body full and it has flavor moderate. It has maker Elyse and it located in the Napa County.

MANUAL-SP: This is a kantharos, created during the Hellenistic period and it originates from Amphipolis. Today it is exhibited in the Archaeological Museum of Kavala.

SEMI-AUTO-SP*: This is a kantharos, produced during the Hellenistic period and almost certainly from Amphipolis. It is found in the Archaeological Museum of Kavala.

SEMI-AUTO-BOOT: This is a kantharos, produced during the Hellenistic period and almost certainly from Amphipolis. It is among the Archaeological Museum of Kavala.

AUTO-SP*: This is a kantharos that handles from the Hellenistic period and is almost certainly from Amphipolis. It derives from the Archaeological Museum of Kavala.

AUTO-BOOT: This is a kantharos that is the Hellenistic period and is Amphipolis. It is the Archaeological Museum of Kavala.

NO-SP: This is a kantharos. It creation period the Hellenistic period, it original location Amphipolis and it current location the Archaeological Museum of Kavala.

MANUAL-SP: Molluscum contagiosum is a kind of viral infectious disease that affects the skin. It results in infections and its symptom is lesions. It is transmitted by fomites and contact with

the skin, and it is caused by the molluscum contagiosum virus.

SEMI-AUTO-SP*: Molluscum contagiosum is a kind of viral infectious disease that can occur in the skin. Infections are caused by molluscum contagiosum. Molluscum contagiosum often causes lesions. It is transmissible by fomites and contact with the skin, and it is caused by the molluscum contagiosum virus.

SEMI-AUTO-BOOT: Molluscum contagiosum is a kind of viral infectious disease that occurs when the skin. Infections are caused by molluscum contagiosum. Molluscum contagiosum can cause lesions. Fomites and contact with the skin can transmit molluscum contagiosum. Molluscum contagiosum is caused by the molluscum contagiosum virus.

AUTO-SP*: Molluscum contagiosum is a kind of viral infectious disease that is the skin. It is infections. It is lesions. It is fomites and contact with the skin, and it is caused by the molluscum contagiosum virus.

AUTO-BOOT: Molluscum contagiosum is a kind of viral infectious disease that is the skin. It is infections. It is lesions. It is fomites, the molluscum contagiosum virus and contact with the skin.

NO-SP: Molluscum contagiosum is a kind of viral infectious disease and it located in the skin. It results in infections. It has symptom lesions. It transmitted by fomites and contact with the skin, and it has material basis in the molluscum contagiosum virus.

As in the corresponding experiment with NL names (Section 3.4.2.4), we generated $95 \times 6 = 570$ texts from the Wine ontology (this time with six configurations), $49 \times 6 = 294$ texts from the M-PIRO Ontology, and $30 \times 6 = 180$ texts from the Disease ontology.[31] The students were asked to score each text by stating how strongly they agreed or disagreed with statements S_1–S_3 below; the non-redundancy criterion was

[31] Compared to the Wine and Disease ontologies, the M-PIRO Ontology provides fewer seeds, which leads to no sentence plans for some of its relations. For those relations, we relaxed the constraint of Section 3.3.1 that requires templates to be extracted from at least two sentences, but this did not always produce good sentence plans.

not used in this experiment, because all six configurations used the same (manually authored) NL names and interest scores. Otherwise, the experimental setup was the same as in the corresponding experiment with NL names (Section 3.4.2.4).

(S_1) *Sentence fluency*: Each sentence of the text (on its own) is grammatical and sounds natural.

(S_2) *Clarity*: The text is easy to understand, provided that the reader is familiar with the terminology and concepts of the domain (e.g., historical periods, grape varieties, virus names).

(S_3) *Semantic correctness*: The text accurately conveys the information of the message triples.

Tables 3.12–3.14 show the scores of the six configurations of NaturalOWL, averaged over the texts of each ontology. For each criterion, the best scores are shown in bold. We performed ANOVA and post-hoc Tukey tests to check for statistically significant differences ($a = 0.05$) in the six scores of each criterion in each ontology. In each criterion (row), we detected no statistically significant differences between scores marked with the same superscript; all the other differences (in the same row) were statistically significant. A post-hoc power analysis of the ANOVA values resulted in power values greater or equal to 0.95 in all cases. MANUAL-SP was the best overall configuration, as one would expect, but SEMI-AUTO-SP* performed only slightly worse, with no detected statistically significant difference between the two configurations in most cases. The only notable exception was the semantic correctness of the M-PIRO Ontology, where the difference between SEMI-AUTO-SP* and MANUAL-SP was larger, because for some relations SEMI-AUTO-SP* produced sentence plans that did not correctly express the corresponding message triples, because of too few seeds. These findings confirm the conclusion of Section 3.4.3.2 that SP* performs very well in a semi-automatic scenario, where a human inspects and selects among the top-ranked sentence plans. SEMI-AUTO-BOOT performed clearly worse than SEMI-AUTO-SP* in this scenario.

In the fully automatic scenario, with no human selection of sentence plans, AUTO-SP* was overall better than AUTO-BOOT, but still not good enough to be used in practice. The NO-SP configuration, which uses sentence plans constructed by tokenizing the identifiers of the OWL relations, obtained much lower sentence fluency and clarity scores than MANUAL-SP, which shows the importance of sentence plans in the perceived quality of the texts. The semantic correctness scores of NO-SP were also much lower than those of MANUAL-SP in the Wine and M-PIRO ontologies, but the difference was smaller (with no detected statistically significant difference) in the Disease Ontology, because tokenizing the OWL identifiers of the relations of the Disease Ontology (e.g., `has_symptom`, `transmitted_by`) leads to sentences that convey the correct information in most cases, even if the sentences are not particularly fluent and clear. The sentence fluency and clarity scores of NO-SP in the Disease Ontology were also higher, compared to the scores of NO-SP in the other two ontologies, for the same reason.

Criteria	NO-SP	AUTO-BOOT	SEMI-AUTO-BOOT	AUTO-SP*	SEMI-AUTO-SP*	MANUAL-SP
Sentence fluency	2.69	3.67^{1}	4.18^{2}	$3.88^{1,2}$	$\mathbf{4.71}^{3}$	$\mathbf{4.75}^{3}$
Clarity	$3.26^{1,2}$	2.89^{1}	4.28	3.47^{2}	$\mathbf{4.81}^{3}$	$\mathbf{4.90}^{3}$
Semantic correctness	$3.02^{1,2}$	2.71^{1}	4.27	3.21^{2}	$\mathbf{4.77}^{3}$	$\mathbf{4.93}^{3}$

Table 3.12: Human scores for texts generated from the Wine Ontology with different methods to obtain sentence plans.

Criteria	NO-SP	AUTO-BOOT	SEMI-AUTO-BOOT	AUTO-SP*	SEMI-AUTO-SP*	MANUAL-SP
Sentence fluency	1.18	2.73	3.63^{1}	4.08^{1}	$\mathbf{4.67}^{2}$	$\mathbf{4.98}^{2}$
Clarity	2.39	1.67	3.12^{1}	3.28^{1}	$\mathbf{4.65}^{2}$	$\mathbf{5.00}^{2}$
Semantic correctness	2.96^{1}	1.90	2.90^{1}	2.86^{1}	4.08	$\mathbf{5.00}$

Table 3.13: Human scores for texts generated from the M-PIRO Ontology with different methods to obtain sentence plans.

Criteria	NO-SP	AUTO-BOOT	SEMI-AUTO-BOOT	AUTO-SP*	SEMI-AUTO-SP*	MANUAL-SP
Sentence fluency	3.40[1]	3.67[1]	**4.07**[1,2]	3.77[1]	**4.77**[2]	**4.83**[2]
Clarity	3.80[1,2]	2.37[1]	3.13	2.53[2]	**4.73**[3]	**4.83**[3]
Semantic correctness	**4.10**[1,2]	2.40[3]	3.27[1,3]	2.63[2]	**4.57**[2]	**4.83**[2]

Table 3.14: Human scores for texts generated from the Disease Ontology with different methods to obtain sentence plans.

3.4.4 Joint experiments with automatically or semi-automatically produced NL names and sentence plans

In a final set of experiments, we examined the effect of combining our methods that produce NL names and sentence plans. We experimented with four configurations of NaturalOWL. The AUTO configuration produces NL names using the method of Section 3.2; it then uses the most highly ranked NL name of each individual or class to produce seed name pairs, and invokes the SP* method of Section 3.3 to produce sentence plans; it then uses the most highly ranked sentence plan for each relation. SEMI-AUTO also produces NL names using the method of Section 3.2, but a human selects the best NL name of each individual or class among the five most highly ranked ones; the selected NL names are then used to produce seed name pairs, and the SP* method is invoked to produce sentence plans; a human then selects the best sentence plan of each relation among the five most highly ranked ones. The MANUAL configuration uses manually authored NL names and sentence plans. In the VERBALIZER configuration, no NL names and sentence plans are provided to NaturalOWL; hence, acting like a simple verbalizer, NaturalOWL produces NL names and sentence plans by tokenizing the OWL identifiers of individuals, classes, and relations.[32] Furthermore, MANUAL uses manually authored interest scores, VERBALIZER uses no interest scores, whereas AUTO and SEMI-AUTO use interest scores obtained from the (top-ranked or selected) NL names

[32] Again, whenever the ontology provides an `rdfs:label` for an individual, class, or relation, VERBALIZER tokenizes the `rdfs:label`.

using the method of Section 3.2.3. All the other linguistic resources (most notably, text plans to order the message triples) are the same (manually authored) across all four configurations. We did not experiment with the BOOT and SP sentence plan generation methods in this section, since SP* performed overall better in the previous experiments. Below are example texts generated from the three ontologies by the four configurations we considered.

MANUAL: This is a moderate, dry Chenin Blanc. It has a full body. It is made by Foxen in the Santa Barbara County.

SEMI-AUTO: This is a full, dry Chenin Blanc wine. It is moderate. It is made at the Foxen Winery in the Santa Barbara region.

AUTO: This is dry Chenin Blanc and is the full. It is the moderate. It is made at Foxen and it is Santa Barbara.

VERBALIZER: Foxen Chenin Blanc is Chenin Blanc. It has sugar Dry, it has maker Foxen and it has body Full. It located in Santa Barbara Region and it has flavor Moderate.

MANUAL: This is a portrait that portrays Alexander the Great and it was created during the Roman period. It is made of marble and today it is exhibited in the Archaeological Museum of Thassos.

SEMI-AUTO: This is a portrait. It is thought to be Alexander the Great, it is produced during the Roman period and it was all hand carved from marble. It is found in the Archaeological Museum of Thasos.

AUTO: This is a portrait. It is thought to be Alexander the Great, it handles from Roman and it is marble. It derives from the Thasos Archaeological Museum.

VERBALIZER: Exhibit 14 is portrait. It exhibit portrays alexander the great. It creation period roman period. It made of marble. It current location thasos archaelogical.

MANUAL: Ebola hemorrhagic fever is a kind of viral infectious disease. Its symptoms are muscle aches, sore throat, fever, weakness, stomach pain, red eyes, joint pain, vomiting, headaches, rashes, internal and external bleeding, hiccups and diarrhea. It is transmitted by infected medical

equipment, contact with the body fluids of an infected animal, and contaminated fomites and it is caused by Bundibugyo ebolavirus, Cote d'Ivoire ebolavirus, Sudan ebolavirus and Zaire ebolavirus.

SEMI-AUTO: An Ebola hemorrhagic fever is a kind of viral Infectious disease. It often causes a muscle ache, a sore throat, a fever, weakness, stomach pain, a red eye symptom, joint pain, vomiting, a headache, rash, a severe internal bleeding, hiccups and diarrhea. It is transmissible by contaminated medical equipment, the direct contact with infected animals, and the contaminated fomite and it is caused by the species Bundibugyo ebolavirus, the Côte d'Ivoire ebolavirus, the Sudan ebolavirus and the Zaire ebolavirus.

AUTO: An Ebola hemorrhagic fever is viral. It is a muscle aches, contaminated medical equipment, a sore throat, a fever, weakness, stomach pain, a red eye, joint pain, a vomiting, a headache, rash, a content internal bleeding symptom, a hiccups and diarrhea. It is caused by the Bundibugyo ebolavirus, a Côte d'Ivoire ebolavirus, the Sudan ebolavirus and the Zaire ebolavirus.

VERBALIZER: Ebola hemorrhagic fever is a kind of viral infectious disease. It has symptom muscle aches, sore throat, fever, weakness, stomach pain, red eyes, joint pain, vomiting, headache, rash, internal and external bleeding, hiccups and diarrhea. It transmitted by infected medical equipment, contact with the body fluids of an infected animal, and contaminated fomites and it has material basis in Bundibugyo ebolavirus, Cote d'Ivoire ebolavirus, Sudan ebolavirus and Zaire ebolavirus.

Again, we note that some NL names and sentence plans of SEMI-AUTO and AUTO can be easily improved using the Protégé plug-in of NaturalOWL. For example, the sentence plan that is used to report the historical period of the exhibit in the second SEMI-AUTO example above can be easily modified to use the simple past tense ("was produced" instead of "is produced"). Nevertheless, we made no such improvements.

Apart from the configurations of NaturalOWL, the experimental setup was the same as in Sections 3.4.2.4 and 3.4.3.5. We generated $95 \times 4 = 380$ texts from the Wine

ontology (with the four configurations), $49 \times 4 = 196$ texts from the M-PIRO Ontology, and $30 \times 4 = 120$ texts from the Disease ontology. The students were now asked to score each text for sentence fluency, clarity, semantic correctness, and non-redundancy, by stating how strongly they agreed or disagreed with statements S_1–S_4 of Section 3.4.2.4.

Criteria	VERBALIZER	AUTO	SEMI-AUTO	MANUAL
Sentence fluency	2.77	3.90	**4.67**[1]	**4.81**[1]
Clarity	3.57	2.79	**4.87**[1]	**4.93**[1]
Semantic correctness	3.57	2.86	**4.68**[1]	**4.97**[1]
Non-redundancy	3.20[1]	3.47[1]	**4.57**[2]	**4.79**[2]

Table 3.15: Human scores for texts generated from the Wine Ontology with different methods to obtain NL names and sentence plans.

Criteria	VERBALIZER	AUTO	SEMI-AUTO	MANUAL
Sentence fluency	2.10	4.33	**4.73**[1]	**4.96**[1]
Clarity	3.02[1]	3.49[1]	4.43	**5.00**
Semantic correctness	3.33[1]	2.90[1]	3.96	**5.00**
Non-redundancy	2.67	4.16[1]	**4.59**[1,2]	**5.00**[2]

Table 3.16: Human scores for texts generated from the M-PIRO Ontology with different methods to obtain NL names and sentence plans.

Tables 3.15–3.17 show the scores of the four configurations of NaturalOWL, averaged over the texts of each ontology. For each criterion, the best scores are shown in bold. Again, we performed ANOVA and post-hoc Tukey tests to check for statistically significant differences ($a = 0.05$) in the four scores of each criterion in each ontology. In each criterion (row), we detected no statistically significant differences between scores marked with the same superscript; all the other differences (in the same row) were statistically significant. A post-hoc power analysis of the ANOVA values resulted in power values equal to 1.0 in all cases. MANUAL had the best overall scores, as one would

Criteria	VERBALIZER	AUTO	SEMI-AUTO	MANUAL
Sentence fluency	3.20[1]	3.00[1]	**4.33**[2]	**4.86**[2]
Clarity	3.77	2.13	**4.47**[1]	**4.90**[1]
Semantic correctness	3.77[1]	2.43	**4.20**[1,2]	**4.93**[2]
Non-redundancy	**4.20**[1,2]	3.43[1]	**4.37**[2]	**4.70**[2]

Table 3.17: Human scores for texts generated from the Disease Ontology with different methods to obtain NL names and sentence plans.

expect, but the scores of SEMI-AUTO were close, in most cases with no detected statistically significant difference, despite the combined errors of the methods that produce NL names and sentence plans. The biggest difference between SEMI-AUTO and MANUAL was in the semantic correctness criterion of the M-PIRO Ontology. This difference is mostly due to the fact that SEMI-AUTO did not always manage to produce sentence plans to convey correctly the semantics of the corresponding message triples, because of too few seeds, as in the experiments of the previous section. This also affected the clarity score of SEMI-AUTO in the M-PIRO Ontology. The scores of AUTO were much lower, which again indicates that our methods cannot be used in a fully automatic scenario.

We also note that the scores of VERBALIZER were overall much lower than those of MANUAL, which again shows the importance of linguistic resources when generating texts from ontologies. The high non-redundancy score of VERBALIZER in the Disease Ontology is due to the fact that there are very few redundant message triples and no anonymous individuals or classes in the Disease Ontology. Hence, VERBALIZER, which treats all the message triples as important and does not anonymize any individuals or classes performs well in terms of non-redundancy. We made a similar observation in the NL name experiments of Section 3.4.2.4.

3.5 Related work

Simple ontology verbalizers (Cregan et al., 2007; Kaljurand and Fuchs, 2007; Schwitter et al., 2008; Halaschek-Wiener et al., 2008; Schutte, 2009; Power and Third, 2010; Power, 2010; Schwitter, 2010a; Liang et al., 2011b) typically produce texts describing individuals and classes without requiring manually authored domain-dependent linguistic resources. They usually tokenize the OWL identifiers or labels (e.g., `rdfs:label`) of the individuals or classes to obtain NL names and sentence plans. We showed (see Section 2.4) showed that the texts of the SWAT verbalizer (Stevens et al., 2011; Williams et al., 2011), one of the best publicly available verbalizers, are perceived as being of significantly lower quality compared to texts generated by NaturalOWL with domain-dependent linguistic resources; NL names, sentence plans, and text plans were found to contribute most to this difference. Without domain-dependent linguistic resources, NaturalOWL was found to generate texts of the same quality as the SWAT verbalizer. NaturalOWL is based on ideas from ILEX (O'Donnell et al., 2001) and M-PIRO (Isard et al., 2003; Androutsopoulos et al., 2007). Excluding simple verbalizers, it seems to be the only publicly available NLG system for OWL, which is why we based our work on it. Nevertheless, its processing stages and linguistic resources are typical of NLG systems (Reiter and Dale, 2000; Mellish et al., 2006b). Hence, we believe that our work is also applicable, at least in principle, to other NLG systems. For example, ONTOSUM (Bontcheva, 2005), which generates natural language descriptions of individuals, but apparently not classes, from RDF SCHEMA and OWL ontologies, uses similar processing stages, and linguistic resources corresponding to NL names and sentence plans.

Ngonga Ngomo et al. (2013) discuss SPARQL2NL, a system that translates SPARQL queries (and possibly their results) to English. SPARQL2NL uses techniques similar to those of simple ontology verbalizers. To express the RDF triples $\langle S, R, O \rangle$ that are involved in a SPARQL query, it assumes that the labels (e.g., `rdfs:label`, perhaps also identifiers) of the relations are verbs or nouns. SPARQL2NL determines if a relation la-

bel is a verb or noun using hand-crafted rules and the POS tags of the label's synonyms in WordNet (Fellbaum, 1998). It then employs manually authored templates, corresponding to our sentence plans, to express the relation; e.g., the template "*S* writes *O*" is used for a triple involving the relation `write`, since "write" is a verb, but "*S*'s author is *O*" is used for the relation `author`, since "author" is a noun. To express the *S* or *O* of a triple, SPARQL2NL tokenizes the label (or identifier) of the corresponding individual or class, pluralizing the resulting name if it refers to a class.

Ratnaparkhi (2000) aims to express a set of attribute-value pairs as a natural language phrase; for example, the set $\{$*city-from* = Athens, *city-to* = New York, *depart-day* = Wednesday$\}$ can be expressed as "flights from Athens to New York on Wednesday". For training purposes, a parallel corpus containing sets of attribute-value pairs, the corresponding phrases, and their dependency trees is required. A maximum entropy model is trained on the corpus, roughly speaking to be able to estimate the probability of a dependency tree given a set of attribute-value pairs. Then, given an unseen set of attribute-value pairs, multiple alternative dependency trees are constructed in a top-down manner, using beam search and the maximum entropy model to estimate the probabilities of the trees being constructed. The most probable tree that expresses all the attribute-value pairs is eventually chosen, and the corresponding phrase is returned. In later work (Ratnaparkhi, 2002), the generated dependency trees are further altered by a set of hand-crafted rules that add unmentioned attributes, and the trees are also ranked by language models. In our case, where we aim to express multiple message triples $\langle S, R_i, O_i \rangle$ all describing an individual or class *S*, we can think of the message triples as attribute-value pairs $R_i = O_i$. To apply the methods of Ratnaparkhi, however, a parallel training corpus with sets of attribute-value pairs (or message triples) and the corresponding target texts would be needed; and corpora of this kind are difficult to obtain. By contrast, our methods require no parallel corpus and, hence, can be more easily applied to ontologies of new domains. Furthermore, the methods of Ratnaparkhi

aim to produce a single sentence per set of attribute-value pairs, whereas our methods produce linguistic resources (NL names, sentence plans) that are used to generate multi-sentence texts (e.g., our NL names and sentence plans include annotations used in sentence aggregation and referring expression generation).

Angeli et al. (2010) generate multi-sentence texts describing database records. Their methods also require a parallel training corpus consisting of manually authored texts and the database records (and particular record fields) expressed by each text. The generative model of Liang et al. (2009) is applied to the training corpus to align the words of each text to the database records and fields it expresses. Templates are then extracted from the aligned texts, by replacing words aligned to record fields with variables. To generate a new text from a set of database records, the system generates a sequence of phrases. For each phrase, it first decides which records and fields to express, then which templates to generate the phrase with, and finally which template variables to replace by which record fields. These decisions are made either greedily or by sampling probability distributions learnt during the training phase. This process is repeated until all the given record fields have been expressed. A language model is also employed to ensure that the transitions between phrases sound natural. As with the work of Ratnaparkhi, the methods of Angeli et al. could in principle be applied to express message triples describing an individual or class, if we think of message triples as record fields, but again a parallel training corpus containing texts and the database records and fields expressed by each text would be needed.

Wong and Mooney (2006; 2007) employ Statistical Machine Translation (SMT) methods to automatically obtain formal semantic representations from natural language sentences. They automatically construct a synchronous context-free grammar, by applying a statistical word alignment model to a parallel training corpus of sentences and their semantic representations. The grammar generates both natural language sentences and their semantic representations. Given a new sentence, the grammar produces can-

didate semantic representations, and a maximum-entropy model (trained on the parallel corpus) estimates the probability of each candidate representation. Chen and Mooney (2008) use the same methods in the reverse direction, to convert formal semantic representations to single sentences. In principle, similar SMT methods could be employed to generate sentences from message triples (or OWL statements). However, a parallel corpus of texts and message triples would again be needed. Furthermore, SMT methods produce a single sentence at a time, whereas our work concerns generating multi-sentence texts.

Lu et al. (2009) generate natural language sentences from tree-structured semantic representations (Lu et al., 2008). Given a parallel training corpus of sentences and their tree-structured semantic representations, hybrid trees are created by expanding the original semantic representation trees of the corpus with nodes standing for phrases of the corresponding sentences. To generate a new sentence from a tree-structured semantic representation, a set of candidate hybrid trees is initially produced based on predefined tree patterns and a Conditional Random Fields model trained on the hybrid trees of the parallel corpus. A sentence is then obtained from the most probable candidate hybrid tree. In later work, Lu and Ng (2011) extend their hybrid trees to support formal logic (typed lambda calculus) semantic representations. A synchronous context free grammar is obtained from the extended hybrid trees of the parallel corpus. The grammar is then used to map formal logic expressions to new sentences. We note that OWL is based on description logic (Baader et al., 2002) and, hence, methods similar to those of Lu et al. could in principle be used to map OWL statements to sentences, though the hybrid trees would have to be modified for description logic. A parallel training corpus of texts and description logic expressions (or corresponding OWL statements) would again be needed, however, and only single sentences would be obtained.

Konstas and Lapata (2012b) use a probabilistic context-free grammar to convert a given set of database entries to a single sentence (or phrase). Roughly speaking, in

each parse tree of the grammar, the leaves are the words of a sentence, and the internal nodes indicate which database entries are expressed by each subtree. The grammar is constructed using hand-crafted templates of rewrite rules and a parallel training corpus of database entries and corresponding sentences; a generative model based on the work of Liang et al. (2009) is employed to estimate the probabilities of the grammar rules. Subsequently, all the parse trees of the grammar for the sentences of the training corpus and the corresponding database entries are represented as a weighted directed hypergraph (packed forest) (Klein and Manning, 2002). The hypergraph's weights are estimated using the inside-outside algorithm (Li and Eisner, 2009) on the training corpus. Following the algorithm of Huang and Chiang (2007), the hypergraph nodes are then integrated with an n-gram language model trained on the sentences of the corpus. Given a new set of database entries, the most probable derivation is found in the hypergraph using a k-best Viterbi search with cube pruning (Chiang, 2007) and the final sentence is obtained from the derivation. In later work, Konstas and Lapata ((2012a)) find the most probable derivation in the hypergraph by forest reranking, using features that include the decoding probability of the derivation according to their previous work, the frequency of rewrite rules in the derivation, as well as lexical (e.g., word n-grams) and structural features (e.g., n-grams of record fields). The weights of the features are estimated with a structured perceptron (Collins, 2002) on the training corpus.

Apart from simple verbalizers and the work of Ngonga Ngomo et al., all the other related methods discussed above require a parallel training corpus of texts (or sentences, or phrases) and their semantic representations (or trees that include their semantic representations), unlike our work. We have already noted that corpora of this kind are difficult to obtain. A further difference from our work is that all the previous methods assume that the English names of the various entities (or individuals and classes) are already available in the semantic representations of the texts to be generated, or that they can be directly obtained from the identifiers (or labels) of the entities in the semantic

representations. By contrast, we have also proposed methods to produce appropriate NL names for individuals and classes, and we have shown experimentally (Section 3.4.2.4) that without NL names the perceived quality of the generated texts is significantly lower.

Our sentence plan generation method contains a template extraction stage (Section 3.3.1), which is similar to methods that have been proposed to automatically obtain templates that extract instances of particular relations from texts. We discussed bootstrapping template extraction in Section 3.4.3.4. Xu et al. (2007) adopt a similar approach. Given seed pairs of entity names for which a particular relation holds, they find sentences containing the seed pairs, they construct the dependency trees of the sentences, and then produce templates corresponding to the minimum dependency subtrees that cover the entity names of the seed pairs, replacing the entity names by variables. The templates are then ranked using the frequencies of their words in documents of relevant and irrelevant domains, and the best templates are employed to obtain more pairs of entity names for which the relation holds. Similar bootstrapping approaches have been used to obtain paraphrasing (e.g., "X bought Y" $\leftrightarrow$ "Y was acquired by X") and textual entailment rules (e.g., "X bought Y" $\rightarrow$ "Y owns X"); see, for example, the work of Szpektor et al. (2004) and the survey of Androutsopoulos and Malakasiotis (2010). The reader is reminded that the sentence plans we produce are not just templates (e.g., "X bought Y"), but include additional annotations (e.g., POS tags, agreement, voice, tense, cases). Furthermore, they are not intended to capture *all* the alternative natural language expressions that convey a particular relation, unlike information extraction, paraphrase and textual entailment recognition; our goal is to obtain a single sentence plan per relation that leads to high quality texts.

Bootstrapping approaches have also been used to obtain templates that can extract named entities of a particular semantic class (e.g., person names) from texts (Riloff, 1996; Patwardhan and Riloff, 2006). Methods of this kind aim to extract *all* the named entities of a particular class from a corpus. By contrast, we aim to assign a single high

quality NL name to each individual or class of a given ontology. Furthermore, our NL names are not simply strings, but contain additional information (e.g., head, gender, number, agreement) that helps produce high quality texts.

3.6 Summary and Contributions of this Chapter

Concept-to-text generation systems typically require domain-specific linguistic resources to produce high quality texts, but manually constructing these resources can be tedious and costly. We proposed methods that can be used to automatically or semi-automatically extract from the Web sentence plans and natural language names, two of the most important types of domain-specific linguistic resources used by NaturalOWL.

We showed experimentally that texts generated using linguistic resources produced by our methods in a semi-automatic manner, with minimal human involvement, are perceived as being almost as good as texts generated using manually authored linguistic resources, and much better than texts produced by using linguistic resources extracted from the relation and entity identifiers of the ontology. The manual effort to semi-automatically construct the resources was a few hours at most per ontology, while manually authoring the same resources is typically a matter of several days. Also, no familiarity with OWL and the inner workings of NaturalOWL are needed for an end-user to produce natural language names and sentence plans using the methods of this part of the thesis. Unlike previous work, our methods do not require a parallel training corpus of texts and semantic representations, which is particularly important, given that corpora of this kind are very difficult to obtain. Our methods, however, did not perform sufficiently well in a fully-automatic scenario, with no human involvement during the construction of the linguistic resources.

The methods of this part of the thesis have also been embedded in the new version of NaturalOWL. The processing stages and linguistic resources of NaturalOWL are typ-

ical of NLG systems. Hence, we believe that our work is also applicable, at least in principle, to other NLG systems. Our methods may also be useful in simpler ontology verbalizers, where the main concern seems to be to avoid manually authoring domain-specific linguistic resources, currently at the expense of producing texts of much lower quality.

Chapter 4

Generating texts with Integer Linear Programming[1]

4.1 Introduction

Natural Language Generation (NLG) systems typically employ a pipeline architecture (Reiter and Dale, 2000). They usually start their processing by selecting the logical facts (message triples, in the case of NaturalOWL) that will be expressed in the text to be generated. The purpose of the next stage, text planning, ranges from simply ordering the facts to be expressed, in effect also ordering the sentences that will express them, to making more complex decisions about the rhetorical structure of the text. Lexicalization then selects the words and syntactic structures that will realize each fact, specifying how each fact can be expressed as a single sentence. Sentence aggregation may then combine shorter sentences to form longer ones. Another component generates appropriate referring expressions (pronouns, noun phrases etc.), and surface realization produces the final text, based on internal representations of the previous decisions.

[1] Part of the work of this chapter has already been published (Lampouras and Androutsopoulos, 2013b; Lampouras and Androutsopoulos, 2013a).

Each stage of the pipeline is treated as a local optimization problem, where the decisions of the previous stages cannot be modified. This arrangement has engineering advantages (e.g., it is easier to specify the input and output of each stage), but produces texts that may not be optimal, since the decisions of the stages have been shown to be co-dependent (Danlos, 1984; Marciniak and Strube, 2005; Belz, 2008). For example, decisions made during content selection may maximize the total importance of the facts that will be expressed, but the selected facts may be difficult to turn into a coherent text during text planning (e.g., they may be facts about very different aspects of an entity being described); also, content selection and lexicalization may lead to more or fewer sentence aggregation opportunities (e.g., sentences with the same or different verbs). Some of these problems can be addressed by over-generating at each stage (e.g., producing several alternative sets of facts at the end of content selection, several alternative lexicalizations etc.) and employing a final ranking component to select the best combination (Walker et al., 2001). This over-generate and rank approach, however, may also fail to find an optimal solution, and it generates an exponentially large number of candidate solutions when several components are pipelined.

In this chapter, we present an Integer Linear Programming (ILP) model that combines content selection, lexicalization, sentence aggregation, and a limited form of referring expression generation. Figure 4.1 illustrates the main decisions of our ILP model. For content selection, our model decides which of the available facts (meaning message triples about the individual or class to be described) should be expressed. In terms of lexicalization, it decides which sentence plan should be used for each fact that will be expressed, assuming that multiple sentence plans are available per fact. For aggregation, it decides which simple sentences (each reporting a single fact) should be aggregated to form longer sentences, by partitioning the simple sentences (or equivalently the message triples they express) into groups (shown as buckets in Fig. 4.1). After using the ILP model, the aggregation rules of NaturalOWL are applied separately (and greedily,

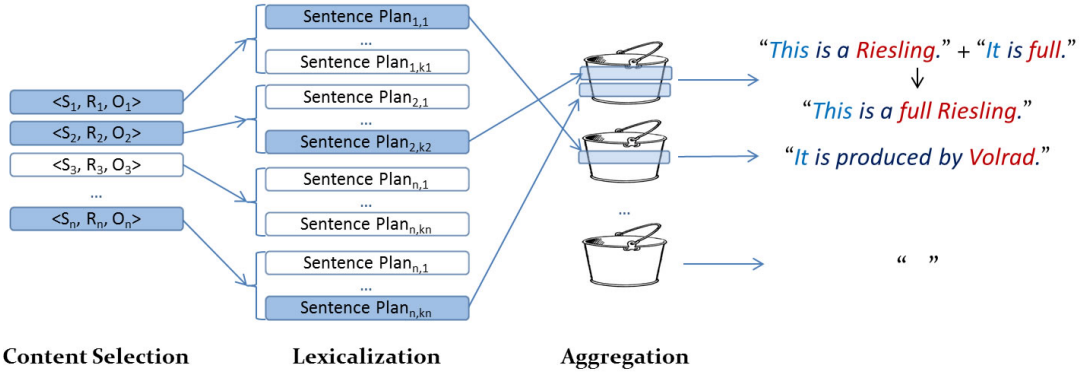


Figure 4.1: Illustration of the main decisions of our ILP model.

as in Section 2.3.2.2) to the simple sentences of each group (bucket) to obtain a single aggregated sentence per group. To keep the ILP model simpler, the model itself does not control which particular aggregation rules will be applied to each group; the ILP model decides only which sentences should be aggregated together. The number of groups (buckets) is fixed, equal to the maximum number of (aggregated) sentences that the model can generate per text. To avoid generating very long aggregated sentences, the number of simple sentences that can be placed in each group (bucket) cannot exceed a fixed upper limit (the same for all groups). Groups left empty produce no sentences.

Our model does not directly consider text planning, but relies on the (external to the ILP model) text planner of NaturalOWL (Section 2.3.1.2). The text planner is invoked before using the ILP model, to partition the available message triples (the triples about the individual or class to be described) into topical 'sections'; this step is needed, because our ILP model never aggregates sentences expressing facts from different sections, to avoid producing aggregated sentences that sound unnatural. The text planner is also invoked after using the ILP model, to order each group of simple sentences that the ILP model has decided to aggregate. As already noted, each aggregated sentence is produced by applying the aggregation rules of Section 2.3.2.2 to the corresponding group of simple sentences, but the rules presuppose that the simple sentences to be ag-

gregated are already ordered, which is why the text planer is invoked at this point. After applying the aggregation rules to each group of (ordered) simple sentences, the text planner is also used to order the sections, and the (now aggregated) sentences within each section.

In terms of referring expression generation, our ILP model selects among multiple alternative NL names of individuals and classes (Section 2.3.2.1), also taking into account that using a particular NL name (e.g., referring to a class as "Bancroft Chardonnay") may make expressing some other facts (e.g., that Bancroft Chardonnay is a kind of Chardonnay) unnecessary (Section 3.2.3). Our ILP model, however, does not consider other decisions of referring expression generation (e.g., deciding to use a pronoun or a demonstrative noun phrase like "this wine", as opposed to repeating the name of a wine). Our model selects a single NL name for each individual or class that will be mentioned in the text to be generated, unless the individual or class is marked as anonymous (Section 2.3.2.3). The existing referring expression generation component of NaturalOWL (Section 2.3.2.3) is then invoked (after the ILP model) to decide if the selected NL name, a pronoun, or a demonstrative noun phrase will be used whenever a reference to an individual or class is needed. We hope to include more text planning and referring expression generation decisions directly in our ILP model in future work. We also do not consider surface realization in this chapter, since it is not particularly interesting in NaturalOWL; all the decisions have already been made by the time we reach this stage.

Unlike pipeline architectures, our ILP model jointly examines the possible choices in the NLG stages it considers, to avoid greedy local decisions.[2] Given an individual or class of an OWL ontology and a set of available facts (message triples) about it, we aim to produce a compact text, meaning a text that expresses (a) as many of the available

[2]The only exception is the application of the aggregation rules to each group (bucket) of simple sentences, which is greedy, as already noted.

facts as possible in (b) as few words as possible. Compact texts of this kind are desirable when space is limited or expensive, for example when displaying product descriptions on smartphones, or when including advertisements in Web search results (Thomaidou et al., 2013; Thomaidou, 2014). By varying weights associated with goals (a) and (b), we obtain different compact texts, aimed towards expressing more of the available facts at the expense of possibly using more words, or aimed towards using fewer words at the expense of possibly expressing fewer of the available facts. If an importance score is available for each fact, our model can take it into account to maximize the total importance (instead of the total number) of the expressed facts in goal (a). The model itself, however, does not produce importance scores, i.e., we assume that the scores are produced by a separate process (Barzilay and Lapata, 2005; Demir et al., 2010), not included in our content selection. For simplicity, in the experiments of this chapter, we treat all the facts as equally important. We also use the terms 'fact' and 'message triple' as synonyms in the remainder of this chapter.

Although the search space of our model is very large and solving ILP problems is in general NP-hard (Karp, 1972), off-the-shelf ILP solvers can be used. The available solvers guarantee finding a globally optimum solution, and they are very fast in practice in the ILP problems we consider, when the the number of available facts (per individual or class being described) is small. We also present an approximation of our ILP model, which is more efficient when the number of available facts is larger. Experiments with three ontologies show that our ILP model (or its approximation) outperforms, in terms of number of expressed facts per word (intuitively, in terms of compression rate), the NaturalOWL system that uses the same components connected in a pipeline, with no deterioration in perceived text quality; the ILP model may actually lead to texts of higher quality, compared to those of the pipeline, in texts that express many facts.

The remainder of this chapter is structured as follows. Section 4.2 discusses previous related work. Section 4.3 defines a first version of our ILP model, which does

not include decisions about NL names. Section 4.4 defines an extended version of our ILP model, which also selects among multiple alternative NL names. Section 4.5 discusses the computational complexity of our ILP models, along with the more efficient approximation that can be used when then number of available facts is large. Section 4.6 presents our experiments. Section 4.7 concludes and summarizes the research contribution of this chapter.

4.2 Related work

Marciniak and Strube (2005) propose a general ILP approach for language processing applications where the decisions of classifiers that consider particular, but co-dependent, subtasks need to be combined. They also show how their approach can be used to generate multi-sentence route directions, in a setting with very different inputs and processing stages than the ones we consider.

Barzilay and Lapata (2005) treat content selection as an optimization problem. Given a pool of facts (database entries) and scores indicating the importance of including or excluding each fact or pair of facts, they select the facts to express by formulating an optimization problem similar to energy minimization. The problem is solved by applying a minimal cut partition algorithm to a graph representing the pool of facts and the importance scores. The importance scores of the facts are obtained via supervised machine learning (AdaBoost) from a dataset of (sports) facts and news articles expressing them. The importance scores of pairs of facts depend on parameters tuned using Simulated Annealing and the same dataset.

In other work, Barzilay and Lapata (2006) consider sentence aggregation. Given a set of facts (again database entries) that a content selection stage has produced, aggregation is viewed as the problem of partitioning the facts into optimal subsets. Sentences expressing facts of the same subset are aggregated to form a longer sentence. The op-

timal partitioning maximizes the pairwise similarity of the facts in each subset, subject to constraints that limit the number of subsets and the number of facts in each subset. A Maximum Entropy classifier predicts the semantic similarity of each pair of facts, and an ILP model is used to find the optimal partitioning.

Althaus et al. (2004) show that the ordering of a set of sentences to maximize local (sentence-to-sentence) coherence is equivalent to the traveling salesman problem and, hence, NP-complete. They also show an ILP formulation of the problem, which can be solved efficiently in practice using branch-and-cut with cutting planes.

Kuznetsova et al. (2012) use ILP to generate image captions. They train classifiers to detect the objects in each image. Having identified the objects of a given image, they retrieve phrases from the captions of a corpus of images, focusing on the captions of objects that are similar (color, texture, shape) to the ones in the given image. To select which objects of the image to report and in what order, Kuznetsova et al. maximize (via ILP) the mean of the confidence scores of the object detection classifiers and the sum of the co-occurrence probabilities of the objects that will be reported in adjacent positions in the caption. The co-occurrence probabilities are estimated from a corpus of captions. Having decided which objects to report and their order, Kuznetsova et al. use a second ILP model to decide which phrases to use for each object and to order the phrases. The second ILP model maximizes the confidence of the phrase retrieval algorithm and the local cohesion between subsequent phrases.

Joint optimization ILP models have also been used in multi-document text summarization and sentence compression (McDonald, 2007; Clarke and Lapata, 2008; Berg-Kirkpatrick et al., 2011; Galanis et al., 2012; Woodsend and Lapata, 2012), where the input is text, not formal knowledge representations. Statistical methods to jointly perform content selection, lexicalization, and surface realization have also been proposed in NLG (Liang et al., 2009; Konstas and Lapata, 2012a; Konstas and Lapata, 2012b), but they are currently limited to generating single sentences from flat records, as op-

posed to ontologies. The work of this chapter is the first to consider content selection, lexicalization, and sentence aggregation (and limited referring expression generation) as an ILP joint optimization problem in multi-sentence concept-to-text generation.

4.3 The first version of our ILP model

In this section, we define the first version of our ILP model.[3] This model assumes that there is a single NL name per individual and class (excluding anonymous ones). Hence, no decisions are needed to select among alternative NL names. Furthermore, the model assumes that all the NL names are short and have approximately the same length; hence, the length of a sentence that reports a single fact can be approximately estimated by counting the number of slots (Section 2.3.2.1) of the sentence plan that produced it. These assumptions will be removed in the more elaborate ILP model of the next section.

Let $F = \{f_1, \ldots, f_n\}$ be the set of all the available facts f_i about the individual or class to be described. Recall that we use the term 'fact' as a synonym of 'message triple', and that a message triple has the form $\langle S, R, O \rangle$, where S is an individual or class, O is (ignoring some details) another individual, class, or datatype value, and R is a relation (property) that connects S to O (see Section 2.3.1.1 for more details). To minimize the effect of the (external to the ILP model) text planning and referring expression generation components of NaturalOWL, we set the fact distance (Section 2.3.1.1) to 1 in the work of this chapter, i.e., the S of all the message triples is the individual or class the text is generated for. This way we avoid, for example, NOCB transitions (Section 2.3.1.2) that might otherwise (when the fact distance is 2) be introduced by the text planner and might influence the perceived quality of the generated texts without being directly relevant to the study of the ILP model of this chapter. We also avoid, for exam-

[3] This is the model that was used in the previously published work of this chapter (Lampouras and Androutsopoulos, 2013b; Lampouras and Androutsopoulos, 2013a).

ple, cases where a pronoun intended to refer to a second-level target (Section 2.3.1.1) is misunderstood as referring to the individual or class the text is generated for.

For each fact f_i, we assume that a set $P_i = \{p_{i1}, p_{i2}, \dots\}$ of alternative sentence plans is available. The reader is reminded that each sentence plan p_{ik} specifies how to express $f_i = \langle S_i, R_i, O_i \rangle$ as an alternative single sentence. A sentence plan is a sequence of slots, along with instructions specifying how to fill the slots in. For example, <`:exhibit12`, `:foundIn`, `:athens`> could be expressed using a sentence plan like $[ref(S)]^1$ $[\text{find}]^2_{past,\, passive}$ $[\text{in}]^3$ $[ref(O)]^4$, where square brackets denote slots, $[ref(S)]^1$ and $[ref(O)]^4$ are instructions requiring referring expressions for S and O in the corresponding slots, and $[\text{find}]^2_{past,\, passive}$ requires the passive simple past form of "find". In our example, the sentence plan would lead to a sentence like "Exhibit 12 was found in Athens". We call *elements* the unordered slots with their instructions, but with S and O accompanied by the individuals, classes, or datatype values they refer to; in our example, there are four elements: $[ref(S =$ `:exhibit12`$)]$, $[\text{find}]_{past,\, passive}$, [in], $[ref(O =$ `:athens`$)]$. When all the NL names are short and approximately equally long, the length of a sentence that reports a single fact (message triple) can be approximately estimated by counting the elements of the sentence plan that produced it. Furthermore, by counting the *distinct* elements (no duplicates) of the sentence plans that were used to produce the simple sentences (each reporting a single fact) of a group of sentences to be aggregated (bucket of Fig. 4.1), we can approximately estimate the length of the resulting aggregated sentence, because duplicate elements (originating from more than one simple sentences) are typically expressed only once in the aggregated sentence.

Using the shortest (in elements) sentence plan that can express each selected fact does not necessarily lead to the shortest possible text, because different combinations of sentence plans may lead to more or fewer aggregation opportunities; for example, sentences with the same verb are easier to aggregate. We use the aggregation rules of Section 2.3.2.2, which usually lead to shorter texts, as in the example below.

Bancroft Chardonnay is a kind of Chardonnay. It is made in Bancroft. ⇒ Bancroft Chardonnay is a kind of Chardonnay made in Bancroft.

Content selection also affects aggregation, because the selected facts may or may not have combinations of sentence plans that provide aggregation opportunities (e.g., common verbs), and the aggregation opportunities may allow saving fewer or more words. For example, consider the following facts concerning `MountadamRiesling` and assume that we want to generate a text expressing only four of them.

```
<:MountadamRiesling, isA, :Riesling>
<:MountadamRiesling, :hasFlavor, :Delicate>
<:MountadamRiesling, :hasBody, :Medium>
<:MountadamRiesling, :hasMaker, :Mountadam>
<:MountadamRiesling, :hasSugar, :Dry>
```

A pipeline approach to generation (as in the NaturalOWL version of Chapter 2.3), where the content selection decisions are made greedily and do not consider their effect on the later stage of aggregation, would possibly select the first four of the facts shown above (they could be the most important ones), leading to the following sentences.

This is a medium Riesling and it has delicate flavor. It is produced by Mountadam.

On the other hand, a global approach that considers the decisions of content selection and aggregation jointly would perhaps prefer to express the fifth fact instead of the fourth. In this way, a possibly small decrease in the importance of the facts that are included in the produced text could greatly improve its length, because the sentence plans of the selected facts could aggregate better, as in the example sentence bellow.

This is a medium dry delicate Riesling.

We can now define more formally the first version of our ILP model. Let $s_1, \ldots, s_m$ be disjoint subsets of F (corresponding to the buckets of Fig. 4.1), each containing 0 to n facts, with $m < n$. A single aggregated sentence is generated from each subset s_j

by aggregating the simple sentences (more precisely, their selected sentence plans) that express the facts of s_j.[4] An empty s_j generates no sentence. Hence, the resulting text can be at most m aggregated sentences long. Let us also define:

$$a_i = \begin{cases} 1, & \text{if fact } f_i \text{ is selected} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$l_{ikj} = \begin{cases} 1, & \text{if sentence plan } p_{ik} \text{ is used to express} \\ & \text{fact } f_i, \text{ and } f_i \text{ is in subset } s_j \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$b_{tj} = \begin{cases} 1, & \text{if element } e_t \text{ is used in subset } s_j \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

and let B be the set of all the distinct elements (no duplicates) from all the available sentence plans that can express the facts of F. As already noted, the length of an aggregated sentence resulting from a subset s_j can be roughly estimated by counting the distinct elements of the sentence plans that have been chosen to express the facts of s_j; elements that occur more than once in the chosen sentence plans of s_j are counted only once, because they will probably be expressed only once, due to aggregation.

Our objective function (Eq. 4.4 below) maximizes the total importance of the selected facts (or simply the number of selected facts, if all facts are equally important), and minimizes the number of distinct elements in each subset s_j, i.e., the approximate length of the corresponding aggregated sentence; an alternative explanation is that by minimizing the number of distinct elements in each s_j, we favor subsets that aggregate

[4] All the sentences of every possible subset s_j can be aggregated, because the fact distance is set to 1 in this chapter and, hence, all the sentences share the same subject, which is the class or individual the text is generated for. If the fact distance is set to 2, it may not be possible to aggregate all the sentences of a subset s_j, in which case a subset s_j may correspond to more than one sentences in the final text.

well. By a and b we jointly denote all the a_i and b_{tj} variables. $|\sigma|$ denotes the cardinality of a set σ. The two parts of the objective function are normalized to $[0,1]$ by dividing by the total number of available facts $|F|$ and the number of subsets m times the total number of distinct elements $|B|$. We assume the importance scores $imp(f_i)$ are provided by a separate component (Barzilay and Lapata, 2005; Demir et al., 2010) and range in $[0,1]$. We multiply α_i with the importance score of the corresponding fact. The parameters λ_1, λ_2 are used to tune the priority given to expressing many important facts vs. generating shorter texts; we set $\lambda_1 + \lambda_2 = 1$.

$$\max_{a,b} \lambda_1 \cdot \sum_{i=1}^{|F|} \frac{a_i \cdot imp(f_i)}{|F|} - \lambda_2 \cdot \sum_{j=1}^{m} \sum_{t=1}^{|B|} \frac{b_{tj}}{m \cdot |B|} \tag{4.4}$$

subject to:

$$a_i = \sum_{j=1}^{m} \sum_{k=1}^{|P_i|} l_{ikj}, \text{for } i = 1,\ldots,n \tag{4.5}$$

$$\sum_{e_t \in B_{ik}} b_{tj} \geq |B_{ik}| \cdot l_{ikj}, \text{for} \begin{cases} i = 1,\ldots,n \\ j = 1,\ldots,m \\ k = 1,\ldots,|P_i| \end{cases} \tag{4.6}$$

$$\sum_{p_{ik} \in P(e_t)} l_{ikj} \geq b_{tj}, \text{for} \begin{cases} t = 1,\ldots,|B| \\ j = 1,\ldots,m \end{cases} \tag{4.7}$$

$$\sum_{t=1}^{|B|} b_{tj} \leq B_{max}, \text{for } j = 1,\ldots,m \tag{4.8}$$

$$\sum_{k=1}^{|P_i|} l_{ikj} + \sum_{k'=1}^{|P_{i'}|} l_{i'k'j} \leq 1, \text{for} \begin{cases} j = 1,\ldots,m,\ i = 2,\ldots,n \\ i' = 1,\ldots,n-1; i \neq i' \\ section(f_i) \neq section(f'_i) \end{cases} \tag{4.9}$$

Constraint 4.5 ensures that for each selected fact, only one sentence plan in only one subset is selected; if a fact is not selected, no sentence plan for the fact is selected

either. In Constraint 4.6, B_{ik} is the set of distinct elements e_t of the sentence plan p_{ik}. This constraint ensures that if p_{ik} is selected in a subset s_j, then all the elements of p_{ik} are also present in s_j. If p_{ik} is not selected in s_j, then some of its elements may still be present in s_j, if they appear in another selected sentence plan of s_j.

In Constraint 4.7, $P(e_t)$ is the set of sentence plans that contain element e_t. If e_t is used in a subset s_j, then at least one of the sentence plans of $P(e_t)$ must also be selected in s_j. If e_t is not used in s_j, then no sentence plan of $P(e_t)$ may be selected in s_j. Constraint 4.8 limits the number of elements that a subset s_j can contain to a maximum allowed number B_{max}, in effect limiting the maximum length of an aggregated sentence. Without this constraint, the model might place all the facts in a single subset.

As already noted, for text planning purposes we assume that each relation (property) R has been manually mapped to a single topical 'section'; e.g., relations expressing the color, body, and flavor of a wine may be grouped in one section, and relations about the wine's producer in another (see Section 2.3.1.2). The section of a fact $f_i = \langle S_i, R_i, O_i \rangle$ is the section of its relation R_i. Constraint 4.9 ensures that facts from different sections will not be placed in the same subset s_j, to avoid unnatural aggregations.

4.4 The extended version or our ILP model

The ILP model of the previous section assumes that a single NL name is available for each individual or class (excluding anonymous ones) and, hence, it does not try to select among alternative NL names. By contrast, in this section we assume that multiple alternative NL names are available for each individual or class. The reader is reminded that an NL name specifies how to generate a noun phrase naming the S or O of a fact $\langle S, R, O \rangle$ in a sentence. An NL name is a sequence of slots, along with instructions specifying how to fill the slots in. In the case of `:RedWine`, for example, an NL name like $[]^1_{article,\ indef}$ [red]^2 [wine]^3 could be used, where $[]^1_{article,\ indef}$ produces an indefinite arti-

cle, [red]^2 produces the corresponding adjective, and [wine]^3 the corresponding noun; for simplicity, we ignore some details of NL names in this chapter (see Section 2.3.2.1 for details). In our example, the NL name would be realized as "a red wine".

For the O of each fact $\langle S, R, O \rangle$ to be expressed, we assume that the shortest available NL name is always selected. Hence, we do not model the selection of NL names for Os in the extended ILP model of this section. We take, however, into account the length of the (shortest) NL name of O when estimating the length of a sentence that will be generated to express a fact $\langle S, R, O \rangle$. By contrast, the model of the previous section ignored the lengths of the NL names when estimating sentence lengths, assuming that all the NL names are short and have approximately the same length, an assumption that does not always hold. For example, the Disease Ontology (Section 2.4.3) includes an individual with an NL name that produces "paralysis of the legs due to thrombosis of spinal arteries", and another individual with an NL name producing simply "inflammation". Hence, a sentence that uses the former NL name to express a fact whose O is the former individual will be much longer than a sentence that uses the latter NL name to express another fact whose O is the latter individual, even if both sentences are produced by the same sentence plan.

In the extended model, we consider the possibility that O has the form `and(...)` (see Section 2.3.1.1) that combines multiple classes, individuals, or datatype values.[5] Such a case is demonstrated in the facts below.

```
<:BrazilianHemorrhagicFever, :isA, :ViralInfectiousDisease>
<:BrazilianHemorrhagicFever, :hasMaterialBasisIn, :SabiaVirus>
<:BrazilianHemorrhagicFever, :transmittedBy, :rodents>
<:BrazilianHemorrhagicFever, :hasSymptom, and(:fatigue, :muscleAches, :dizziness)>
```

In the ILP model of the previous section, we made no distinction between Os that are single classes, individuals, or datatype values, and Os that have the form `and(...)`, assuming that the number of conjuncts of each `and(...)` is always small and does not

[5]The case where O has the form `or(...)` is treated in a similar manner.

affect much the length of the resulting sentence. In some ontologies, though, the number of conjuncts can vary greatly; for example, in our OWL version of the Disease Ontology the number of conjuncts in the value of the `hasSymptom` property ranges from 1 to 14. Assuming that we want to generate a text for `BrazilianHemorrhagicFever` and we are limited to expressing two facts, the model of the previous section might, for example, select to express the first and fourth of the facts shown above, possibly because their sentence plans are shorter, leading to the following sentence.

> The Brazilian hemorrhagic fever is a viral infectious disease that causes fatigue, muscle aches and dizziness.

By contrast, the extended ILP model that we define in this section takes into account that the conjunction in the O of the fourth fact above requires five words. Hence, it might select the first and third facts instead, producing the following shorter sentence.

> The Brazilian hemorrhagic fever is a viral infectious disease transmitted by rodents.

Note, also, that selecting the first and second facts, which only have single individuals or classes as Os, would lead to the following sentence, which is longer, because of the length of "the Sabia virus".

> The Brazilian hemorrhagic fever is a viral infectious disease caused by the Sabia virus.

Selecting among the alternative NL names of the S of a fact $\langle S, R, O \rangle$ is more complicated, because a longer NL name (e.g., an NL name that produces "the Napa County Bancroft Chardonay wine") may also convey some of the other facts that can be expressed, without requiring separate sentences for those facts, thus saving words. Consider, for example, the following facts concerning `BancroftChardonnay` and assume that we want to generate a text expressing all these facts.

```
<:BancroftChardonnay, isA, :Chardonnay>
<:BancroftChardonnay, :locatedIn, :NapaRegion>
<:BancroftChardonnay, :hasMaker, :Bancroft>
<:BancroftChardonnay, :hasFlavor, :Moderate>
<:BancroftChardonnay, :hasSugar, :Dry>
```

Let us also assume, for the purposes of our example, that `BancroftChardonnay` has three alternative NL names that produce "Bancroft Chardonnay", "the Napa County Bancroft Chardonnay wine", and "the moderate tasting and dry Bancroft Chardonnay wine", respectively. Some of the NL names of this chapter, like the first two of this example, were automatically generated by the methods of Chapter 3.2.1. The other NL names, like the third one of this example, were manually authored to provide more choices to the extended ILP model.

For each alternative NL name, we apply the method of Section 3.2.3, which assigns zero interest scores to facts conveyed by NL names, to figure out which facts do not need to be expressed by separate sentences.[6] In our example, if we choose to refer to S as "Bancroft Chardonnay", we do not need to produce separate sentences for the first and third facts above, since they are already indirectly expressed by the NL name of S, and similarly for the other two NL names of S, as shown below.

S called "Bancroft Chardonnay":

Bancroft Chardonnay is moderate and dry. It is produced in the Napa County.

~~It is a Chardonnay. It is produced by Bancroft.~~

S called "the Napa County Bancroft Chardonnay wine":

The Napa County Bancroft Chardonnay wine is moderate and dry.

~~It is a Chardonnay. It is produced by Bancroft in the Napa County.~~

[6] When using the method of Section 3.2.3 to determine if a fact $\langle S, R, O \rangle$ does not need to be expressed by a separate sentence if a particular NL name of S is selected, we consider all the alternative NL names of O, not just the shortest one.

S called "the moderate tasting and dry Bancroft Chardonnay wine":

The moderate tasting and dry Bancroft Chardonnay wine is produced in the Napa County.

~~It is a moderate, dry Chardonnay. It is produced by Bancroft.~~

Selecting the NL name that produces the shortest noun phrase ("Bancroft Chardonnay") does not lead to the shortest text. The shortest text is obtained when the second NL name is selected. Selecting the NL name (the third one above) that leads to the largest number of facts made redundant (meaning facts that no longer need to be expressed as separate sentences) also does not lead to the shortest text, as shown above.

To further increase the range of options that the extended ILP model considers, and possibly allow it to produce more compact texts, when using the extended ILP model we allow alternative NL names to be provided also for individuals or classes declared as anonymous (Section 2.3.2.3). When an anonymous S of a fact $\langle S, R, O \rangle$ has alternative NL names, we allow the extended ILP model to use either a demonstrative "this" or any of the alternative NL names to refer to S.[7]

Using a demonstrative may lead to a shorter text in some cases, but not always. Continuing our example, the following text uses a demonstrative to refer to the wine being described.

Demonstrative used for *S*:

This is a moderate, dry Chardonnay. It is produced by Bancroft in the Napa County.

Before moving on to the formulation of the extended ILP model, let us discuss how it measures the lengths of sentence plans and NL names. In the ILP model of the previous section, we roughly estimated the length of an aggregated sentence resulting from a subset (bucket) s_j by counting the distinct elements of the sentence plans chosen to

[7] A demonstrative that includes an ancestor class (e.g., "this statue") can also be produced, as in Section 2.3.2.3, in which case the demonstrative is also taken to express the corresponding fact about the ancestor class (e.g., `<:exhibit31, isA, :Statue>`).

express the facts of s_j. For example, let us assume that the distinct elements [*ref*(*S* = `:exhibit12`)], [find]$_{past,\ passive}$, [in], and [*ref*(*O* = `:athens`)] are used in a single subset s_j. The ILP model of the previous section did not consider the lengths of the noun phrases that will be produced by the NL names of `:exhibit12` and `:athens` of the elements [*ref*(*S* = `:exhibit12`)] and [*ref*(*O* = `:athens`)]. It also did not take into account that the element [find]$_{past,\ passive}$ actually produces two words ("was found").

The extended model of this section defines a function $length(e_t)$ that maps each distinct element e_t to the length (in words) of the text it produces (e.g., "was found"). More specifically, if e_t is an element referring to O (e.g., [*ref*(*O* = `:athens`)]), then $length(e_t)$ is the length of the (shortest) NL name of O; if O has the form `and(...)`, then $length(e_t)$ is the sum of the lengths of the (shortest) NL names of all the conjuncts. However, if e_t is an element referring to S (e.g., [*ref*(*S* = `:exhibit12`)]), then $length(e_t) = 1$, because the NL name of S will be used only once at the beginning of the text, and each subsequent reference to S will be via a pronoun of length 1 (e.g., "Bancroft Chardonnay is moderate and dry. *It* is produced in the Napa County."); the first realization of the NL name is counted separately, directly in the objective function discussed below. The estimated length of a (possibly aggregated) sentence is the sum of the estimated lengths (in words) of the distinct elements of the sentence plan(s) that produced it. Overall, the extended model estimates more accurately the length of the text that will be produced, though the actual text length may still be slightly different; for example, connectives (e.g., "and", "or") may be added during aggregation.

We can now formally define the extended version of our ILP model. Let $N = \{n_1, n_2, \dots\}$ be a set of alternative NL names for the class or individual S we generate a text for. As already discussed, we model only the choice of NL name for S, assuming that the shortest NL name is always used for the O_i of each fact $f_i = \langle S, R_i, O_i \rangle$. As in the simpler model of Section 4.3, F is the set of available facts f_i about S, and $s_1, \dots, s_m$ are disjoint subsets of F (buckets of Fig. 4.1) showing which simple sentences (each ex-

pressing a single fact of F) will be aggregated together. Each a_i variable now indicates if the corresponding fact f_i is explicitly expressed by generating a sentence:

$$a_i = \begin{cases} 1, & \text{if the fact } f_i \text{ is expressed as a sentence} \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

By contrast, d_i is more general; $d_i = 1$ if the corresponding fact f_i is conveyed either explicitly (by generating a sentence for f_i) or implicitly (via an NL name):

$$d_i = \begin{cases} 1, & \text{if the fact } f_i \text{ is expressed as a sentence or via an NL name} \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

The distinction between a_i and d_i is necessary, because when a fact f_i is explicitly expressed as a sentence in the text, a sentence plan for f_i must also be selected. As an example, a fact f_i = <:BancroftChardonnay, :hasMaker, :Bancroft> can either be expressed as a sentence in the final text (e.g., "This is produced by Bancroft. It comes from the Napa County.") or through an NL name (e.g., "Bancroft Chardonnay is produced in the Napa County."). In both texts the fact f_i is expressed (i.e., $d_i = 1$), but in the former text $a_i = 1$, whereas in the latter text $a_i = 0$.

The l_{ikj} and b_{tj} variables are as in the ILP model of the previous section (Eq. 4.2 and 4.3). For the extended model, we also define:

$$m_r = \begin{cases} 1, & \text{if the NL name } n_r \text{ is used for } S \\ 0, & \text{otherwise} \end{cases} \tag{4.12}$$

Similarly to the previous model's objective function (4.4), the extended model's objective function (4.13) maximizes the total importance of the expressed facts (or simply the number of expressed facts, if all facts are equally important), and minimizes the length of the distinct elements in each subset s_j and the length of the NL name used to express S, i.e., the approximate length of the corresponding aggregated sentence. By d, b, and m we jointly denote all the d_i, b_{tj}, and m_r variables. The left part of the objective is the same as in the previous model, with the variables a_i replaced by d_i. In the right

part, we multiply the b_{tj} and m_r variables with the functions $length(e_t)$ and $length(n_r)$, which calculate the lengths of the corresponding element and NL name, respectively. The two parts of the objective function are normalized to $[0,1]$ by dividing by the total number of available facts $|F|$ and the number of subsets m times the total length of distinct elements $|B|$ plus the total length of the R available NL names. Again, the parameters λ_1, λ_2 are used to tune the priority given to expressing many important facts vs. generating shorter texts; we set $\lambda_1 + \lambda_2 = 1$.

$$\max_{d,b,m} \lambda_1 \cdot \sum_{i=1}^{|F|} \frac{d_i \cdot imp(f_i)}{|F|} - \lambda_2 \cdot \left(\frac{\sum_{j=1}^{m}\sum_{t=1}^{|B|} b_{tj} \cdot length(e_t) + \sum_{r=1}^{|R|} m_r \cdot length(n_r)}{m \cdot \sum_{t=1}^{|B|} length(e_t) + \sum_{r=1}^{|R|} length(n_r)}\right) \tag{4.13}$$

subject to:

$$a_i = \sum_{j=1}^{m}\sum_{k=1}^{|P_i|} l_{ikj}, \text{for } i = 1,\ldots,n \tag{4.14}$$

$$\sum_{e_t \in B_{ik}} b_{tj} \geq |B_{ik}| \cdot l_{ikj}, \text{for} \begin{cases} i = 1,\ldots,n \\ j = 1,\ldots,m \\ k = 1,\ldots,|P_i| \end{cases} \tag{4.15}$$

$$\sum_{p_{ik} \in P(e_t)} l_{ikj} \geq b_{tj}, \text{for} \begin{cases} t = 1,\ldots,|B| \\ j = 1,\ldots,m \end{cases} \tag{4.16}$$

$$\sum_{t=1}^{|B|} b_{tj} \cdot length(e_t) \leq W_{max}, \text{for } j = 1,\ldots,m \tag{4.17}$$

$$\sum_{k=1}^{|P_i|} l_{ikj} + \sum_{k'=1}^{|P_{i'}|} l_{i'k'j} \leq 1, \text{for} \begin{cases} j = 1,\ldots,m,\ i = 2,\ldots,n \\ i' = 1,\ldots,n-1; i \neq i' \\ section(f_i) \neq section(f'_i) \end{cases} \tag{4.18}$$

$$\sum_{r=1}^{|N|} m_r = 1 \tag{4.19}$$

$$d_i = a_i + \sum_{m_r \in R(f_i)} m_r, \text{for } i = 1,\ldots,n \tag{4.20}$$

Constraints 4.14–4.18 serve the same purpose as in the previous model, with the exception of 4.17, which now limits the number of words (instead of elements) that a subset s_j can use to a maximum allowed number W_{max}. Constraint 4.19 ensures that only one NL name is selected from the available NL names of S. In Constraint 4.20, $R(d_i)$ is the set of NL names that express the fact f_i. If f_i is to be expressed (i.e., $d_i = 1$), then either one of the NL names in $R(d_i)$ must be selected, or a sentence for f_i must be generated ($a_i = 1$). If f_i is not to be expressed, then none of the NL names in $R(d_i)$ may be selected, nor should a sentence be generated for f_i.

4.5 Computational complexity and approximations

The models we presented in Sections 4.3 and 4.4 are formulated as ILP problems, and more precisely as binary ILP problems since all their variables are binary. Solving binary ILP problems is in general NP-hard (Karp, 1972). Even otherwise formulated, our models perform content selection amongst available weighted facts into a distinct number of subsets, similarly to the 0-1 multiple Knapsack problem, which is NP-hard. However, our ILP models also consider interactions between the selected facts (e.g., during aggregation) and are, therefore, further constrained.

A practical approach to solve ILP models in polynomial time is to relax the constraint that all variables are integer (or binary) and solve the resulting Linear Programming model (LP relaxation) using, for example, the Simplex algorithm (Dantzig, 1963). The resulting values of the variables are then rounded to the closest integral values. The resulting solution is not guaranteed to be optimal for the original ILP problem, nor feasible (some constraints of the original problem may be violated). The solution of the LP relaxation is guaranteed to be the same as the solution of the original ILP problem, though, if the problem can be formulated as $\max_x c^T x$ with constraints $Ax = b$, where c,

A, and m have integer values and the matrix A is totally unimodular (Schrijver, 1986). An integer matrix is totally unimodular if every square, nonsingular submatrix is unimodular (i.e., its determinant is 0, 1, or -1). Unfortunately, this is not the case in our ILP models.

In practice, off-the-shelf ILP solvers are very fast when the number of variables is small. The experiments of the Section 4.6.2 below show that solving the first ILP model is reasonably fast, provided that the number of fact subsets (buckets, maximum number of aggregated sentences of the final text) does not exceed 4 ($m \leq 4$). The number of subsets (m) seems to be the greatest factor to the model's complexity; the number of variables in the model grows exponentially to the number of fact subsets, while the effect of the other parameters (e.g., number of available facts $|F|$) is not so great. We did not perform separate experiments examining how the solving times of the extended ILP model relate to the number of subsets m, however the critical value $m \leq 4$ should hold, since the variables in the extended model are similar and also grow exponentially to the number of fact subsets m.

When the number of variables is too large to solve the first ILP model reasonably fast, we use an approximation of the model, which considers each fact subset (bucket, i.e., aggregated sentence of the final text) separately. By treating each fact subset on its own (with $m = 1$), in effect we treat the 0-1 multiple Knapsack problem as a list of distinct 0-1 Knapsack problems. Figure 4.2 illustrates the approximation of the first ILP model. We start with the full set of available facts (F) and use the ILP model with $m = 1$ to optimally produce the first (aggregated) sentence of the text. We then remove the facts expressed by the first (aggregated) sentence from F, and use the ILP model, again with $m = 1$, to produce the second (aggregated) sentence etc. This process is repeated until we produce the maximum number of allowed aggregated sentences, or until we run out of available facts.

Since the approximation of the first ILP model does not consider all the fact sub-

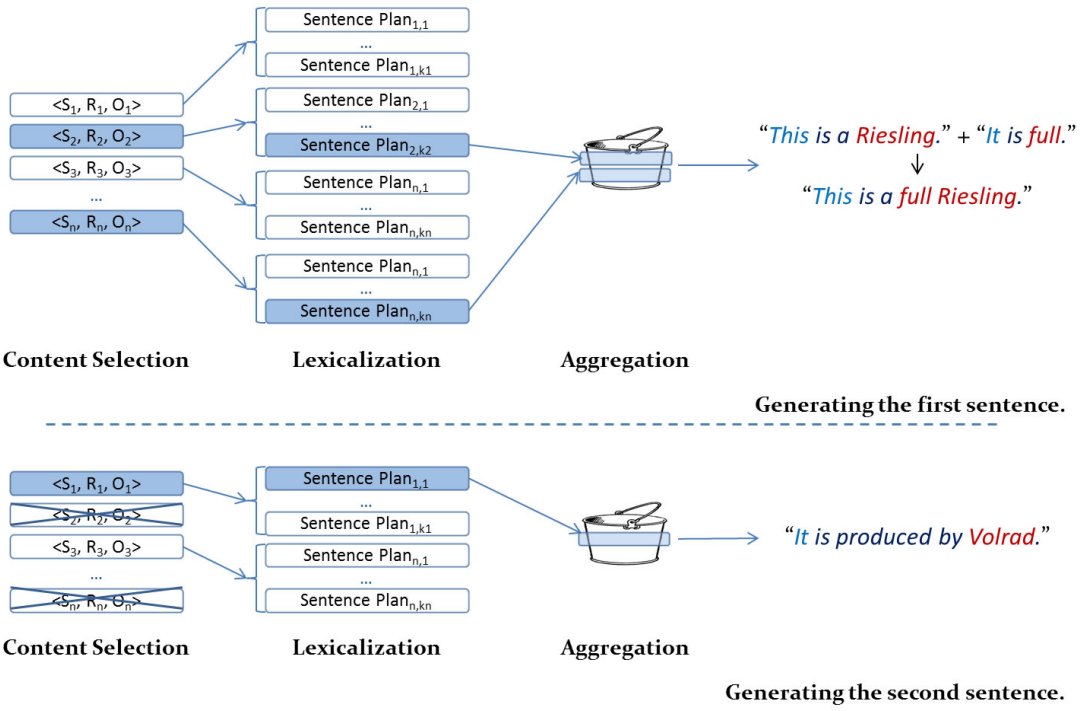


Figure 4.2: Illustration of the approximation of the first ILP model.

sets jointly, it does not guarantee finding a globally optimal solution for the entire text. Despite this, experiments (in Section 4.6.2 below) that compare the approximation to the original first ILP model show no apparent decline in text quality nor in the ability to produce compact texts. Solving times now grow almost linearly to the number of subsets m and the number of available facts $|F|$. Furthermore, $|F|$ decreases in every subsequent solving of the model (to produce the next aggregated sentence of the text), which reduces the time needed by the solver. Experiments indicate that the approximation can guarantee practical running times even for $m \geq 5$, while still outperforming the pipeline approach in terms of producing more compact texts.

The same approximation (considering each fact subset separately) can be applied to our extended ILP model. We did not experiment with the approximation of the extended model, however, because the only ontology we considered that required $m \geq 5$ and, hence, an approximation (Consumer Electronics, Section 4.6.2 below) did not require

using the extended model (the lengths of the NL names did not vary significantly, and we could not think of alternative NL names for the products being described).

4.6 Experiments

We call PIPELINE the NaturalOWL version of Chapter 2.3, which uses a pipeline architecture. We constructed two more systems, called ILPNLG and ILPNLGEXTEND, where the content selection, lexicalization, and aggregation components of PIPELINE were replaced by a component that implements our first (Section 4.3) or extended (Section 4.4) ILP model. We use branch-and-cut to solve the ILP problems.[8] We set the importance scores $imp(f_i)$ of all the facts f_i to 1, to make the decisions of all the systems we experimented with easier to understand; all systems use the same importance scores and provide no mechanism to estimate them, assuming that they are provided manually.

For referring expression generation and surface realization, ILPNLG and ILPNLGEXTEND invoke the same components as PIPELINE. For aggregation, all three systems use the same set of aggregation rules. In PIPELINE, the rules are applied greedily (as in Section 2.3.2.2) to all the selected facts, after invoking the text planner (Section 2.3.1.2) to order the sections and the selected facts in each section. In ILPNLG and ILPNLGEXTEND, the text planner is first invoked to order the selected facts of each subset (bucket), then the aggregation rules are applied separately to each subset, greedily inside each subset, and then the text planner is invoked to order the sections and the (now aggregated) sentences inside each section.[9]

PIPELINE has a parameter M specifying the maximum number of facts it is allowed to report per text (see Section 2.3.2.2). In effect, PIPELINE aims to report M facts

[8] We use the branch-and-cut implementation of GLPK with mixed integer rounding, mixed cover, and clique cuts; see `http://sourceforge.net/projects/winglpk/`.

[9] In each section, every aggregated sentence inherits the minimum order (Section 2.3.1.2) of the properties of the facts it expresses.

per text. When M is larger than the number of available facts ($|F|$), it reports all the available facts. When M is smaller than the number of available facts and all the facts are treated as equally important, it selects randomly M of the available facts. In the experiments that follow, we generated texts with PIPELINE for different values of M. For each M, the texts of PIPELINE for each class or individual were generated three times, each time using a different, randomly selected, alternative sentence plan of each relation and a different (randomly selected) NL name of each individual or class (when multiple alternative NL names were available). For each M, the reported results of PIPELINE are averaged over all the generated texts. We also generated the texts (for different values of M) using a variant of PIPELINE, dubbed PIPELINESHORT, which always selects the shortest (in elements) sentence plan among the available ones and the shortest (in words) NL name.

PIPELINE, PIPELINESHORT, and ILPNLG assume that there is a single NL name for each individual or class (excluding anonymous ones). Another variant of PIPELINESHORT, dubbed PIPELINESHORT*, always selects the shortest (now in words) sentence plan among the available ones, and the NL name of S that indirectly expresses the largest number of available facts $f_i = \langle S, R_i, O_i \rangle$ (thus not requiring sentences to express them).[10] For O_i, PIPELINESHORT* selects the same (shortest) NL name as ILPNLGEXTEND. PIPELINESHORT* is a more appropriate baseline for ILPNLGEXTEND than PIPELINE and PIPELINESHORT, because it allows multiple alternative NL names, and estimates the lengths of the sentences in words, like ILPNLGEXTEND.

In the following subsections, we present the experiments we performed on the Wine, Consumer Electronics, and Disease Ontologies. We started by comparing ILPNLG to PIPELINE and PIPELINESHORT on the Wine Ontology (Section 2.4.1), where experi-

[10] In our experiments, selecting the NL name of S that expresses the largest number of available facts usually leads to better facts per word ratios than simply selecting the shortest (in words) NL name of S. If several NL names of S express the same number of facts, PIPELINESHORT* selects the shortest NL name.

ments showed that ILPNLG leads to more compact texts, i.e., texts whose ratio of expressed facts per word is higher, with no deterioration in the perceived quality of the resulting texts, compared to the texts of PIPELINE and PIPELINESHORT.

We then tried to repeat the same experiments on the Consumer Electronics Ontology, but ILPNLG was too slow to be used in many cases, because of the larger number of available facts per product ($|F|$) and the larger ($m = 10$) number of subsets (buckets) that were required to be able to express all (or many) of the available facts. To address this problem, we developed the approximation (Section 4.5) of ILPNLG, dubbed ILPNLGAPPROX, which achieved greater facts per word ratios than PIPELINE and PIPELINESHORT, with no deterioration in the perceived quality of the texts. When texts expressing many facts had to be generated, the perceived quality of the texts of ILPNLGAPPROX was actually higher, comparing to the texts of PIPELINE and PIPELINESHORT.

We then moved on to the Disease Ontology, to experiment with an additional domain.[11] Since the Disease Ontology only required $m = 4$ fact subsets to express all the available facts per disease, ILPNLGAPPROX was not required, and ILPNLG was used instead. We found that ILPNLG did not always perform better than PIPELINE and PIPELINESHORT (in terms of facts per word ratios) on the Disease Ontology, because the lengths of the NL names of this ontology vary a lot, and there are also several facts $\langle S, R, O \rangle$ whose O is an `and(...)`, sometimes with many conjuncts. To address these issues, we extended ILPNLG to ILPNLGEXTEND, which consistently produced more compact texts than PIPELINE and PIPELINESHORT* on the Disease Ontology.

Lastly, we returned to the Wine Ontology to see how ILPNLGEXTEND performs when multiple alternative NL names are available. For this experiment, we created alternative NL names for the individuals and classes of the Wine Ontology; we could not

[11] The Disease Ontology was also used in the BIOASQ project (`http://www.bioasq.org/`), where there was a need to convert the ontology to high quality, compact English texts, which could then be included in the input of multi-document summarizers.

do the same for the Consumer Electronics and the Disease Ontology, because the names of electronics products tend to be unique and we did not have the expertise to create alternative names of diseases. Experiments verified that ILPNLGEXTEND performs better (in terms of facts per word ratios) than PIPELINE and PIPELINESHORT* on the Wine Ontology, when multiple NL names are available.

4.6.1 Experiments with the Wine Ontology

In a first set of experiments, we used the Wine Ontology, which had also been used in the experiments of Section 2.4.1. We kept the manually authored domain-dependent generation resources of the experiments of Section 2.4.1, including the topical sections, the ordering of sections and relations, the user modelling preferences, and the sentence plans and NL names, but we added more sentence plans to ensure that three sentence plans were available per relation. We generated English texts for the 52 wine individuals of the ontology; we did not experiment with texts describing classes, because we could not think of multiple alternative sentence plans for many of their axioms. For each wine individual, there were 5 facts on average and a maximum of 6 facts.

We generated texts with ILPNLG, PIPELINE, and PIPELINESHORT for the 52 individuals. With PIPELINE and PIPELINESHORT, we generated texts for $M = 2,3,4,5,6$; recall that M is the maximum allowed number of facts per text, in effect the number of facts that PIPELINE and PIPELINESHORT aim to report per text. In all cases, PIPELINE and PIPELINESHORT were allowed to form aggregated sentences with up to $B_{max} = 22$ distinct elements; this was the number of distinct elements of the longest aggregated sentence in the experiments of Section 2.4.1, where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one.[12] With ILPNLG, we repeated the generation of the texts of the 52 individuals using different values of λ_1 ($\lambda_2 = 1 - \lambda_1$), which led to texts expressing from zero to all of the

[12] We modified PIPELINE and PIPELINESHORT to count distinct elements during aggregation.

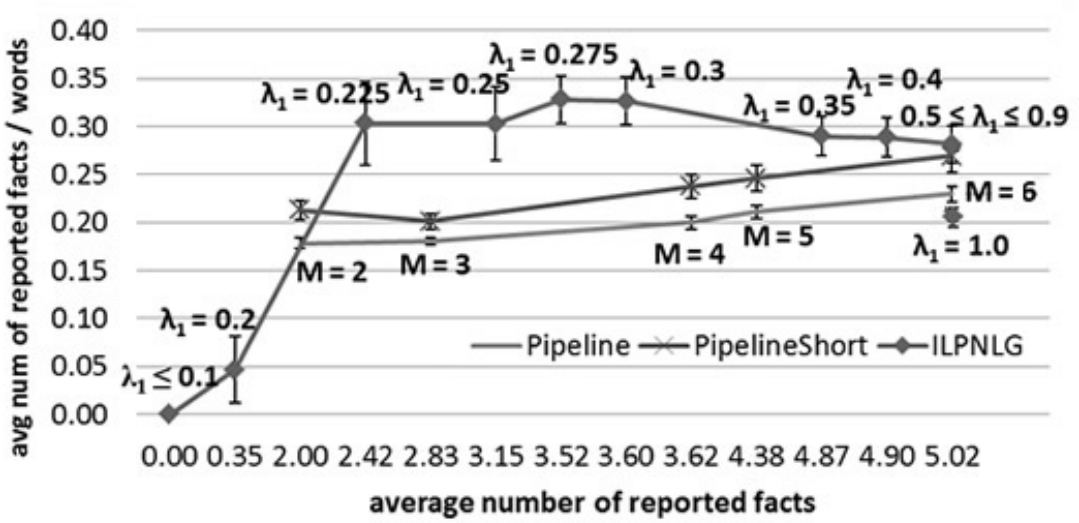


Figure 4.3: Facts per word ratios for the Wine Ontology.

available facts. We set the maximum number of fact subsets (buckets) to $m = 3$, which was the maximum number of sentences (after aggregation) in the texts of PIPELINE and PIPELINESHORT. We allowed ILPNLG to form aggregated sentences with up to $B_{max} = 22$ distinct elements, as with PIPELINE and PIPELINESHORT.

For each M value (in the case of PIPELINE and PIPELINESHORT) and for each λ_1 value (in the case of ILPNLG), we measured the average (over the 52 texts) number of facts each system reported per text (horizontal axis of Fig. 4.3), and the average (again over the 52 texts) number of facts each system reported per text divided by the average (over the 52 texts) number of words (vertical axis of Fig. 4.3, with error bars showing 95% confidence intervals of sample means).[13] As one would expect, PIPELINESHORT expressed on average more facts per word (Fig. 4.3) than PIPELINE, but the differences were small.

For $\lambda_1 < 0.2$ (far left of Fig. 4.3), ILPNLG produces empty texts, because it focuses on minimizing the number of distinct elements of each text. For $\lambda_1 \geq 0.2$, it performs better than PIPELINE and PIPELINESHORT. For $\lambda_1 \approx 0.3$, it obtains the highest average

[13] For $0.5 \leq \lambda_1 \leq 0.9$, ILPNLG's results were identical.

facts per word ratio by selecting the facts and sentence plans that lead to the most compressive aggregations. For greater values of λ_1, it selects additional facts whose sentence plans do not aggregate that well, which is why the ratio declines. When the number of facts to be selected is small, the two pipeline systems often select facts and sentence plans that offer few aggregation opportunities; as the number of selected facts increases, some more aggregation opportunities arise, which is why the facts per word ratio of the two systems improves.

Figure 4.4 provides an alternative view of the behavior of the three systems. In this case, we group together all the texts of each system (regardless of the M or λ_1 values that were used to generate them) that report 2, 3, 4, 5, or 6 facts (horizontal axis of Fig. 4.4). For each group (and each system), we show (vertical axis of Fig. 4.4) the average number of reported facts per text, divided by the average number of words of the texts in the group.[14] Again, Figure 4.4 shows that ILPNLG produces clearly more compact texts than PIPELINE and PIPELINESHORT, with the difference between the latter two systems being very small.[15] In all the experiments of this section, the ILP solver was very fast (average: 0.08 sec, worst: 0.14 sec per text).

We show below texts produced by PIPELINE and PIPELINESHORT (both with $M = 4$) and ILPNLG (with $\lambda_1 = 0.3$).

PIPELINE: This Sauternes has strong flavor. It is made from Sauvignon Blanc grapes and Semil-

[14] We remove from each group duplicate (identical) texts generated by the same system (for different M or λ_1 values) for the same individual or class. If we still have more than one texts (of the same system) for the same individual or class in the same group, we keep only the one with the best facts per word ratio per individual or class, to avoid placing too much emphasis on individuals and classes with many texts in the same group. Especially for PIPELINE, whose texts are generated three times per individual and class (with different sentence plans), we keep the three texts (excluding duplicates) with the highest facts per word ratios per individual or class in each group.

[15] Figure 4.4 and all the similar figures in the remainder of this chapter include error bars corresponding to 95% confidence intervals, but the intervals are so small that they can hardly be seen.

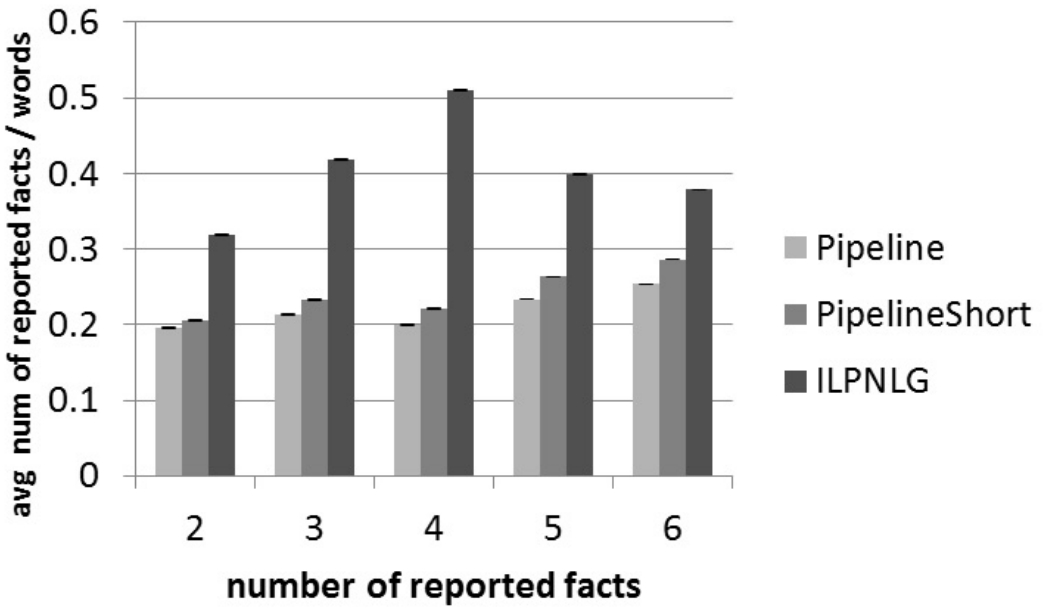


Figure 4.4: Facts per word ratios for the Wine Ontology (grouped by reported facts).

lon grapes and it is produced by Chateau D'ychem.

PIPELINESHORT: This is a strong Sauternes. It is made from Sauvignon Blanc grapes and Semillon grapes and it is produced by Chateau D'ychem.

ILPNLG: This is a strong Sauternes. It is made from Sauvignon Blanc grapes and Semillon grapes by Chateau D'ychem.

PIPELINE: This Riesling has sweet taste and it is full bodied. It is made by Schloss Volrad.

PIPELINESHORT: This is a full sweet Riesling. It is produced by Schloss Volrad.

ILPNLG: This is a full sweet moderate Riesling.

In the first group of generated texts above, PIPELINE and PIPELINESHORT use different verbs for the grapes and producer, whereas ILPNLG uses the same verb, which leads to a more compressive aggregation; all the texts of the first group describe the same wine and report four facts each. In the second group of generated texts above, ILPNLG has chosen to report the (moderate) flavor of the wine instead of the producer, and uses the

same verb ("be") for all the facts, leading to a shorter sentence; again all the texts of the second group describe the same wine and report four facts each. In both groups of texts, some facts are not aggregated because they belong in different topical sections.

We also wanted to investigate the effect that the higher facts per word ratio of ILPNLG has on the perceived quality of the generated texts, compared to the texts of the pipeline systems. We were concerned that the more compressive aggregations of ILPNLG might lead to sentences sounding less fluent or unnatural, though aggregation often helps produce more natural texts. We were also concerned that the more compact texts of ILPNLG might be perceived as being more difficult to understand (less clear) or less well-structured. To investigate these issues, we showed the $52 \times 2 = 104$ texts of PIPELINESHORT ($M = 4$) and ILPNLG ($\lambda_1 = 0.3$) to 6 computer science students (undergraduates and graduates), who were not involved in the work of this thesis; they were all fluent, though not native English speakers. Both PIPELINESHORT and ILPNLG were tuned to report all the available facts per text. We did not use PIPELINE in this experiment, since its facts per word ratio was very similar to that of PIPELINESHORT. Each one of the 104 texts was given to exactly one student. Each student was given approximately 9 randomly selected texts of each system. The OWL statements that the texts were generated from were not shown, and the students did not know which system had generated each text. Each student was shown all of his/her texts in random order, regardless of the system that generated them. The students were asked to score each text by stating how strongly they agreed or disagreed with statements S_1–S_3 below. A scale from 1 to 5 was used (1: strong disagreement, 3: ambivalent, 5: strong agreement).

(S_1) *Sentence fluency*: The sentences of the text are fluent, i.e., each sentence *on its own* is grammatical and sounds natural. When two or more smaller sentences are combined to form a single, longer sentence, the resulting longer sentence is also grammatical and sounds natural.

(S_2) *Text structure*: The order of the sentences is appropriate. The text presents information by moving reasonably from one topic to another.

(S_3) *Clarity*: The text is easy to understand, provided the reader is familiar with basic wine terms.

Criteria	PIPELINESHORT	ILPNLG
Sentence fluency	4.75 ± 0.21	**4.85** ± 0.10
Text structure	**4.94** ± 0.06	4.88 ± 0.14
Clarity	**4.77** ± 0.18	4.75 ± 0.15
Overall	4.52 ± 0.20	**4.60** ± 0.18

Table 4.1: Human scores for Wine Ontology texts.

The students were also asked to provide an overall score (1–5) per text. We did not score referring expressions, since both systems use the same component for them.

Table 4.1 shows the average scores of the two systems with 95% confidence intervals (of sample means). For each criterion, the best score is shown in bold. The sentence fluency and overall scores of ILPNLG are slightly higher than those of PIPELINESHORT, whereas PIPELINESHORT obtained a slightly higher score for text structure and clarity. The differences, however, are very small, especially in clarity, and we detected no statistically significant difference between the two systems in any of the criteria.[16] Hence, there was no evidence in these experiments that the highest facts per word ratio of ILPNLG comes at the expense of lower perceived text quality. We investigated these issues further in a second set of experiments, discussed in the next section, where the generated texts were longer.

4.6.2 Experiments with the Consumer Electronics Ontology

In the second set of experiments, we used the Consumer Electronics Ontology, which had also been used in the experiments of Section 2.4.2. We kept the manually au-

[16] We performed Analysis of Variance (ANOVA) and post-hoc Tukey tests to check for statistically significant differences. A post-hoc power analysis of the ANOVA values resulted in power values greater or equal to 0.95. We also note that in the similar experiments of Section 2.4.1, where judges were asked to score texts using the same criteria, inter-annotator agreement was strong (sample Pearson correlation $r \geq 0.91$).

thored domain-dependent generation resources of Section 2.4.2, including the sections, ordering of sections and relations, user modelling preferences, sentence plans and NL names, but we added more sentence plans (using the 30 development individuals of Section 2.4.2 to preview the resulting texts) to ensure that three sentence plans were available for almost every relation; for some relations we could not think of enough sentence plans.

We generated texts with ILPNLG, PIPELINE, and PIPELINESHORT for the 30 development individuals of Section 2.4.2, using $M = 3,6,9,\ldots,21$ in the two pipeline systems, and different values of λ_1 ($\lambda_2 = 1 - \lambda_1$) in ILPNLG. All three systems were allowed to form aggregated sentences with up to $B_{max} = 39$ distinct elements; this was the number of distinct elements of the longest aggregated sentence in the experiments of Section 2.4.2, where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one. There are 14 available facts ($|F|$) on average and a maximum of 21 facts for each one of the 30 development individuals, compared to the 5 facts on average and the maximum of 6 facts of the experiments with the Wine Ontology. Hence, the texts of the Consumer Electronics Ontology are much longer, when they report all the available facts. To generate texts for the 30 individuals with ILPNLG, we would have to set the maximum number of fact subsets to $m = 10$, which was the maximum number of (aggregated) sentences in the texts of PIPELINE and PIPELINESHORT. However, as discussed in Section4.5, the number of variables of our ILP model grows exponentially to m, and is also affected (though not as strongly in practice) by $|F|$.

Figure 4.5 shows the average time the ILP solver took for different values of m in the experiments with the Consumer Electronics ontology; the results are averaged over the 30 development individuals and also for $\lambda_1 = 0.4, 0.5, 0.6$. For $m = 4$, the solver took 1 minute and 47 seconds on average per text; recall that $|F|$ is also much larger now, compared to the experiments of the previous section. For $m = 5$, the solver was so slow

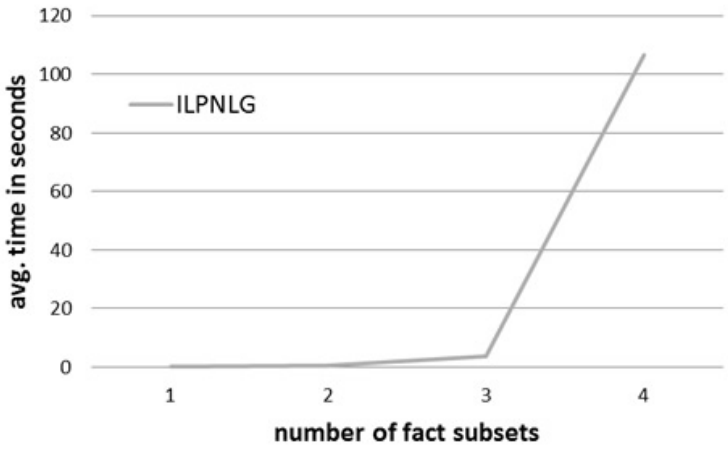


Figure 4.5: Average solver times for ILPNLG with different maximum numbers of fact subsets (m), when generating texts from the Consumer Electronics ontology.

that we aborted the experiment. Figure 4.6 shows the average solver times for different numbers of available facts $|F|$, for $m = 3$; in this case, we modified the set of available facts (F) of every individual to contain $3, 6, 9, 12, 15, 18, 21$ facts. The results are again averaged over the 30 development individuals and for $\lambda_1 = 0.4, 0.5, 0.6$. Although the times of Fig. 4.6 also grow exponentially to $|F|$, they remain under 4 seconds, showing that the main factor to the complexity of ILPNLG is m, the number of fact subsets, i.e., the maximum allowed number of (aggregated) sentences of each text.

To efficiently generate texts with larger m values, we developed ILPNLGAPPROX, which considers each fact subset separately (Section 4.5). Figures 4.7–4.8 show the average solver times of ILPNLGAPPROX for different values of m and $|F|$, respectively; all the other settings are as in Figures 4.5–4.6. The solver times of ILPNLGAPPROX grow approximately linearly to m and $|F|$ and are under 0.3 seconds in all cases.

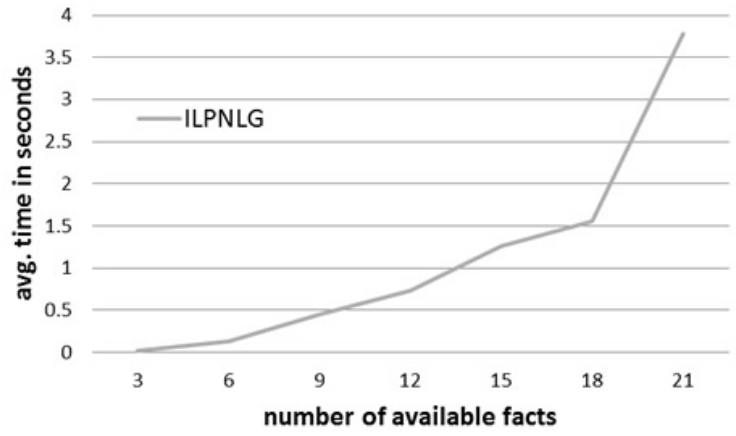


Figure 4.6: Average solver times for ILPNLG with different numbers of available facts ($|F|$) and $m = 3$, when generating texts from the Consumer Electronics ontology.

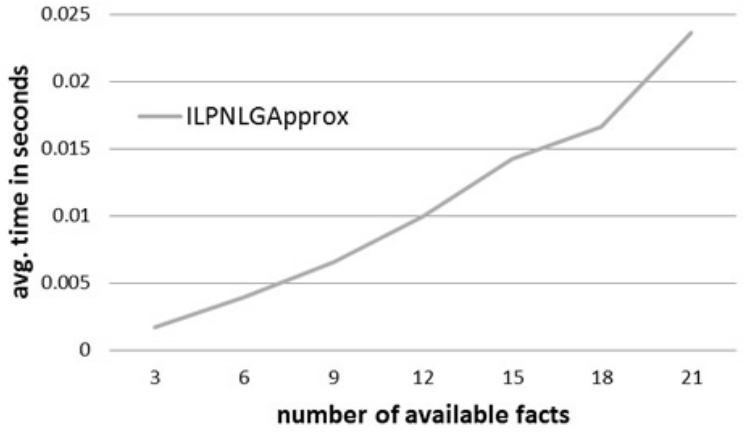


Figure 4.7: Avg. solver times for ILPNLGAPPROX with different numbers of available facts ($|F|$) and $m = 3$, when generating texts from the Consumer Electronics ontology.

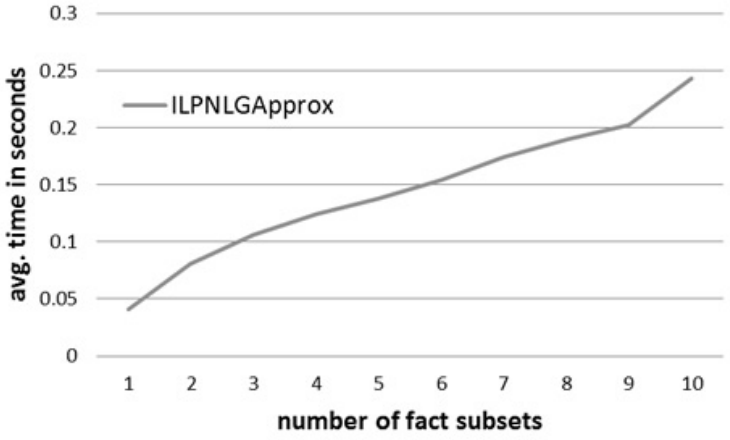


Figure 4.8: Average solver times for ILPNLGAPPROX with different numbers of fact subsets (m), when generating texts from the Consumer Electronics ontology.

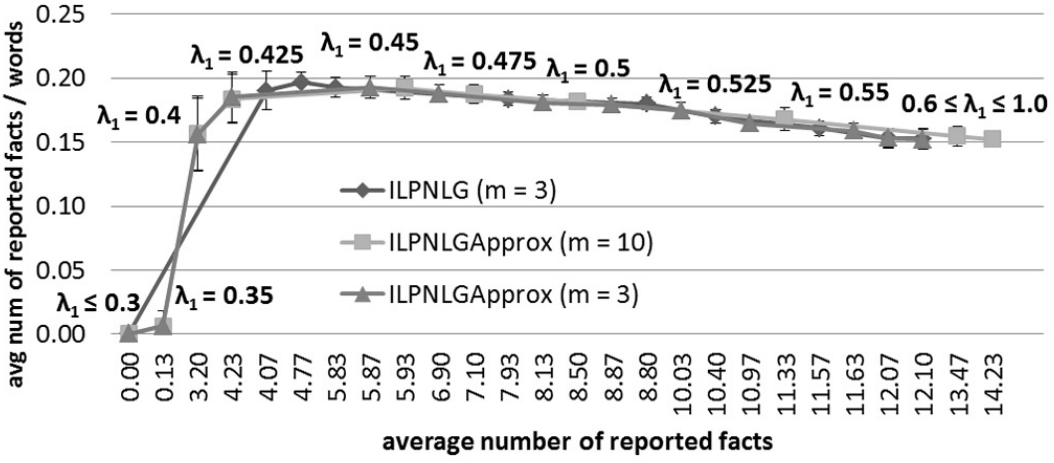


Figure 4.9: Comparing the facts per word ratios of ILPNLGAPPROX and ILPNLG in texts generated from the Consumer Electronics ontology.

In Figure 4.9, we compare ILPNLG to ILPNLGAPPROX, by showing their average fact per word ratios, computed as in Fig. 4.3 (Section 4.6.1). We set $m = 3$ in ILPNLG to keep the solving times low; in ILPNLGAPPROX we experimented with both $m = 3$ (the value used in ILPNLG) and $m = 10$ (the value that was actually needed, as discussed above). In all cases, $B_{max} = 39$. We observe that the facts per word ratios of all three systems are very similar. We conclude that ILPNLGAPPROX achieves very similar results to ILPNLG in much less time.

Figures 4.10 and 4.11 show the facts per word ratios of ILPNLGAPPROX ($m = 10$), PIPELINE, and PIPELINESHORT, computed in two ways, as in Section 4.6.1, for the texts of the 30 development individuals.[17] Again, PIPELINESHORT achieves slightly better results than PIPELINE. The behavior of ILPNLGAPPROX in Figure 4.10 is very similar to the behavior of ILPNLG on the Wine Ontology (Fig. 4.3); for $\lambda_1 \leq 0.3$, it produces empty texts, while for $\lambda_1 \geq 0.4$ it performs better than the other systems. ILPNLGAPPROX obtains the highest facts per word ratio for $\lambda_1 = 0.45$, where it selects the facts and sentence plans that lead to the most compressive aggregations. For greater values of λ_1, it selects additional facts whose sentence plans do not aggregate that well, which is why the ratio declines. The two pipeline systems select facts and sentence plans that offer very few aggregation opportunities; as the number of selected facts increases, some more aggregation opportunities arise, which is why the facts per word ratio of the two systems improves as we move towards the right end of Fig. 4.10 and 4.11. Fig. 4.11 also shows that ILPNLGAPPROX generates more compact texts than PIPELINE and PIPELINESHORT.

We show below three example texts produced by PIPELINE, PIPELINESHORT ($M = 6$), and ILPNLGAPPROX ($\lambda_1 = 0.45, m = 10$). Each text reports six facts, but ILPNLGAPPROX has selected facts and sentence plans that allow more compressive aggregations. Recall that we treat all the facts as equally important. If importance scores are also

[17] For $0.6 \leq \lambda_1 \leq 0.9$, ILPNLGAPPROX's results in Fig. 4.10 were identical.

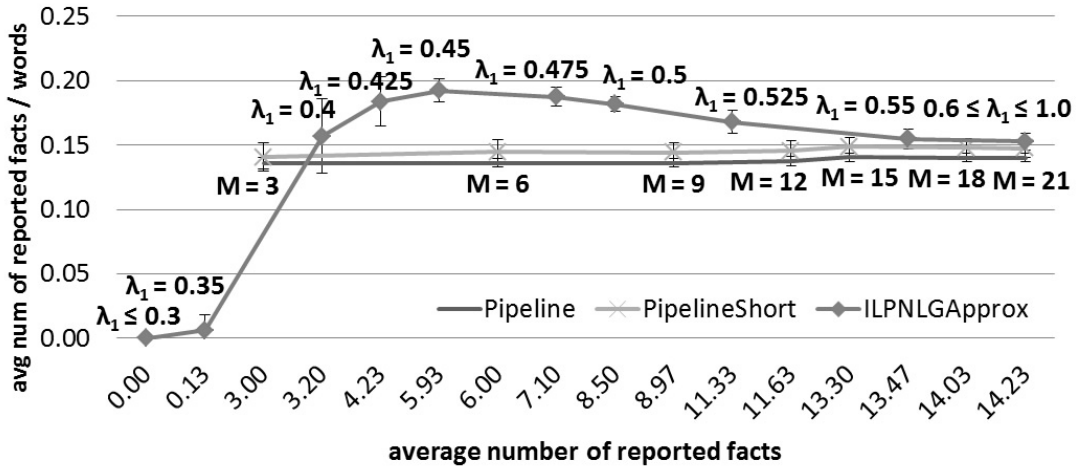


Figure 4.10: Facts per word ratios for the Consumer Electronics Ontology.

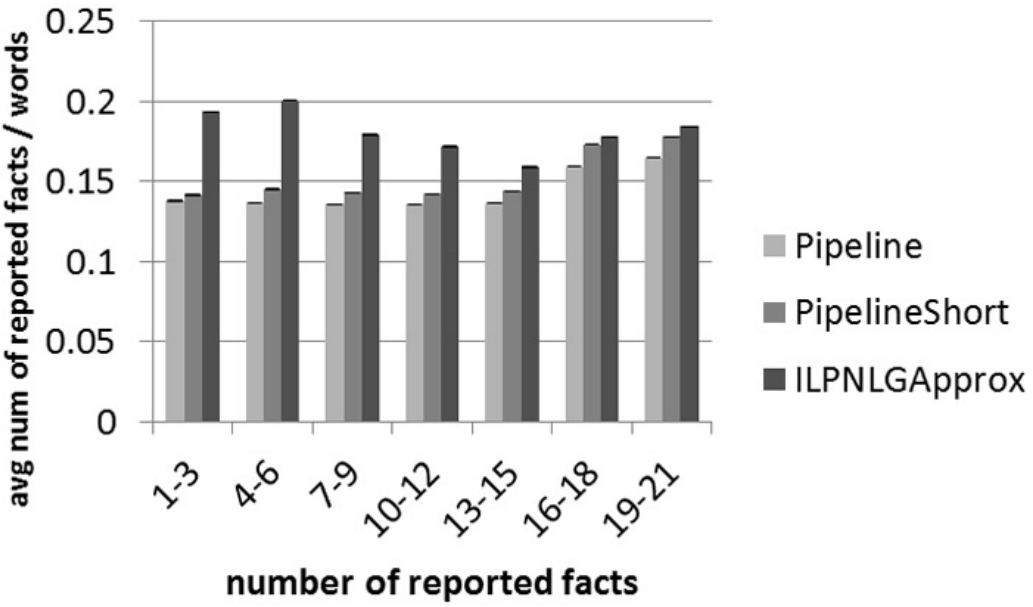


Figure 4.11: Facts per word ratios for the Consumer Electronics Ontology (grouped by reported facts).

available (e.g., if dimensions are less important), they can be added as multipliers of α_i in the objective function (Eq. 4.4) of our ILP model, as noted in Section 4.3.

PIPELINE: SonySony DCR-TRV270 requires minimum illumination of 4.0 lux and its display is 2.5 in. It features a Sports scene mode, it includes a microphone and an IR remote control. Its weight is 780.0 grm.

PIPELINESHORT: Sony DCR-TRV270 requires minimum illumination of 4.0 lux and its display is 2.5 in. It features a Sports scene mode, it includes a microphone and an IR remote control. It weighs 780.0 grm.

ILPNLGAPPROX: Sony DCR-TRV270 has a microphone and an IR remote control. It is 98.0 mm high, 85.0 mm wide, 151.0 mm deep and it weighs 780.0 grm.

We showed the $30 \times 2 = 60$ texts of PIPELINESHORT ($M = 6$) and ILPNLGAPPROX ($\lambda_1 = 0.45$, $m = 10$) to the same six students that participated in the experiments with the Wine Ontology (Section 4.6.1). Both systems were tuned to report all the available facts per text. Again, each text was given to exactly one student. Each student was given approximately 5 randomly selected texts of each system. The OWL statements that the texts were generated from were not shown, and the students did not know which system had generated each text. Each student was shown all of his/her texts in random order, regardless of the system that generated them. The students were asked to score each text by stating how strongly they agreed or disagreed with statements S_1–S_3, as in Section 4.6.1. They were also asked to provide an overall score (1–5) per text.

Table 4.2 shows the average scores of the two systems with 95% confidence intervals (of sample means). For each criterion, the best score is shown in bold; the confidence interval of the best score is also shown in bold, if it does not overlap with the confidence interval of the other system. Unlike the Wine Ontology experiments (Table 4.1), the scores of our ILP approach (with the approximation of ILPNLGAPPROX) are now higher than those of PIPELINESHORT in all of the criteria, and the differences

Criteria	PIPELINESHORT	ILPNLGAPPROX
Sentence fluency	4.50 ± 0.30	$\mathbf{4.87} \pm 0.12$
Text structure	4.33 ± 0.36	$\mathbf{4.73} \pm 0.22$
Clarity	4.53 ± 0.29	$\mathbf{4.97 \pm 0.06}$
Overall	4.10 ± 0.31	$\mathbf{4.73 \pm 0.16}$

Table 4.2: Human scores for Consumer Electronics texts.

are also larger, though we found the differences to be statistically significant only for clarity and overall quality.[18] We attribute these larger differences, comparing to the Wine Ontology experiments, to the fact that the texts are now longer and the sentence plans more varied, which often makes the texts of PIPELINESHORT sound verbose and, hence, more difficult to follow, compared to the more compact texts of ILPNLGAPPROX, which sound more concise.

Overall, the human scores of the experiments with the Wine and Consumer Electronics ontologies suggest that the higher facts per word ratios of our ILP approach do not come at the expense of lower perceived text quality. On the contrary, the texts of the ILP approach may be perceived as clearer and overall better than those of the pipeline, when the texts report many facts.

4.6.3 Experiments with the Disease Ontology

In this third set of experiments, we used the Disease Ontology, which had also been used in the experiments of Section 2.4.3. We kept the manually authored domain-dependent generation resources of Section 2.4.3, including the topical sections, the ordering of sections and relations, user modelling preferences, sentence plans, and NL names, but we

[18] When two confidence intervals do not overlap, the difference is statistically significant. When they overlap, the difference may still be statistically significant; we performed Analysis of Variance (ANOVA) and post-hoc Tukey tests to check for statistically significant differences in those cases. A post-hoc power analysis of the ANOVA values resulted in power values greater or equal to 0.95.

added more sentence plans to ensure that 3 sentence plans were available per relation.

We generated texts with ILPNLG, PIPELINE, and PIPELINESHORT for the 200 development and test classes of Section 2.4.3, using $M = 2,3,4,\ldots,7$ in the two pipeline systems, and different values of λ_1 ($\lambda_2 = 1 - \lambda_1$) in ILPNLG. All three systems were allowed to form aggregated sentences with up to $B_{max} = 30$ distinct elements; this was the number of distinct elements of the longest aggregated sentence in the experiments of Section 2.4.3, where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one. There are 3.7 available facts ($|F|$) on average and a maximum of 7 facts for each one of the 200 development classes. In ILPNLG, we set $m = 4$, which was the maximum number of (aggregated) sentences in the texts of PIPELINE and PIPELINESHORT. We did not use ILPNLGAPPROX in these experiments, since ILPNLG was reasonably fast (average solver time: 0.11 sec per text, worst: 0.90 sec per text), because of the smaller values of m and $|F|$, compared to the experiments of the Consumer Electronics ontology.

Figures 4.12 and 4.13 show the facts per word ratios of ILPNLG, PIPELINE, and PIPELINESHORT, computed in two ways, as in Section 4.6.1, for the texts of the 200 development classes.[19] Again, PIPELINESHORT achieves only slightly better results than PIPELINE in both figures. Also, Fig. 4.13 shows that ILPNLG produces more compact texts than the two pipeline systems. In the view of Figure 4.12, however, the difference between ILPNLG and the two pipeline systems is less clear. For small λ_1 values, ILPNLG produces empty texts, because it focuses on minimizing the number of distinct elements of each text. For $\lambda_1 \geq 0.125$, it performs only marginally better than the pipeline systems, unlike previous experiments (cf. Fig. 4.3 and 4.10). We attribute this difference to the fact that ILPNLG does not take into account the lengths of the NL names, which vary a lot in the Disease Ontology; it also does not take into account that the O of many facts $\langle S,R,O\rangle$ of the Disease Ontology is an `and(...)`. To address these issues, we

[19] For $0.18 \leq \lambda_1 \leq 0.9$, ILPNLG's results in Fig. 4.12 were identical.

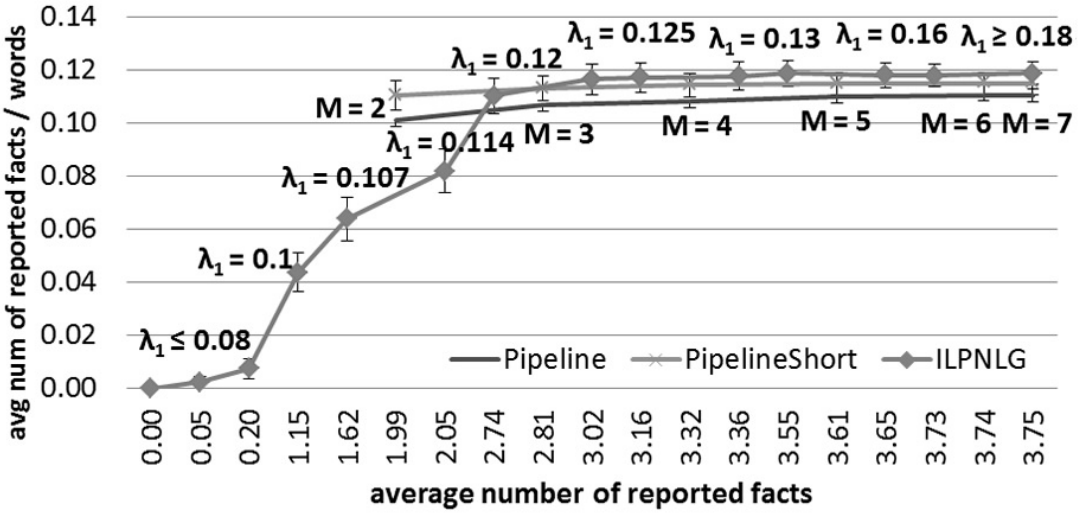


Figure 4.12: Facts per word ratios of ILPNLG, PIPELINE, and PIPELINESHORT for texts generated from the Disease Ontology.

defined our extended ILP model, as already discussed in Section 4.4.

We then generated texts for the 200 development classes again, this time with PIPELINE, PIPELINESHORT* (both with $M = 2,3,4,5,6,7$, $W_{max} = 54$) [20], and ILPNLGEXTEND ($m = 4$, $W_{max} = 54$). Similarly to how B_{max} was selected, $W_{max} = 54$ was the number of words of the longest aggregated sentence in the experiments of Section 2.4.3, where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one. Figures 4.14 and 4.15 show the new facts per word ratios, for the texts of the 200 classes.[21] In Figure 4.14, for $\lambda_1 \leq 0.06$, ILPNLGEXTEND produces empty texts, because it focuses on minimizing the lengths of the texts. For $\lambda_1 \geq 0.12$, ILPNLGEXTEND now performs clearly better than the pipeline systems, obtaining the highest facts per word ratio for $\lambda_1 = 0.14$; notice that we now compare to PIPELINESHORT*, which is a better baseline for ILPNLGEXTEND than PIPELI-

[20] We modified PIPELINE and PIPELINESHORT to count words (instead of elements) during aggregation when comparing against ILPNLGEXTEND.

[21] For $0.4 \leq \lambda_1 \leq 0.9$, ILPNLGAPPROX's results in Fig. 4.14 were identical.

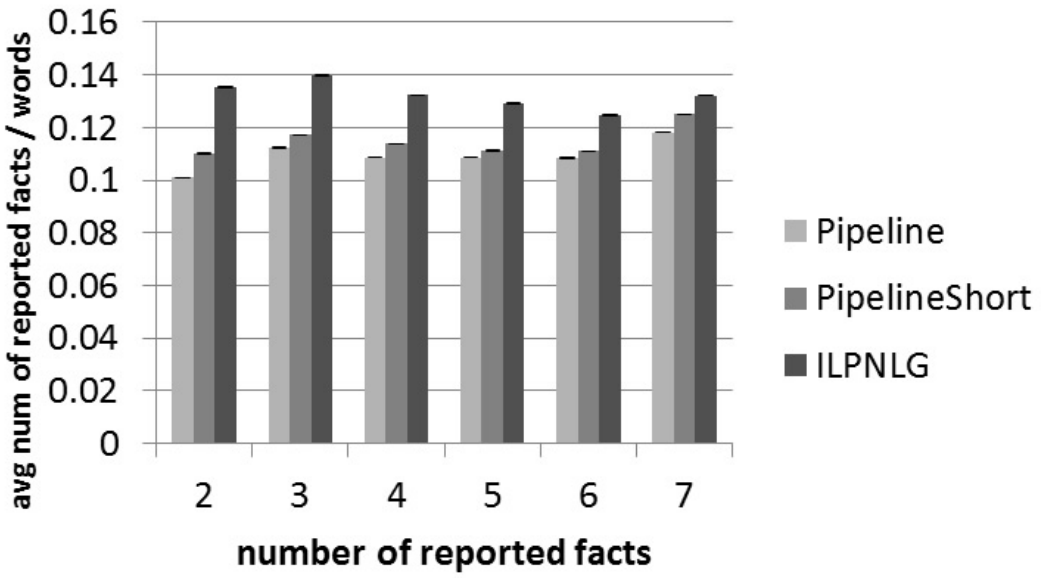


Figure 4.13: Facts per word ratios (grouped by reported facts) of ILPNLG, PIPELINE, and PIPELINESHORT for texts generated from the Disease Ontology.

NESHORT, as already noted. Figure 4.15 also shows that ILPNLGEXTEND outperforms the pipeline systems. The ILP solver was almost as fast with ILPNLGEXTEND (average: 0.09 sec, worst: 0.65 sec per text), as with ILPNLG.

We show below three example texts produced by PIPELINE, PIPELINESHORT* ($M = 3$), and ILPNLGEXTEND ($\lambda_1 = 0.14$). Each text reports three facts, but ILPNLGEXTEND has selected facts with fewer and shorter NL names fewer and shorter NL names.

PIPELINE: Rift valley fever can often cause vomiting blood, bleeding from the nose, jaundice, menorrhagia, passing blood in the feces, ecchymoses (caused by bleeding in the skin) and bleeding from venepuncture sites. It is transmitted by the aedes mosquitoes and it has a material basis in Rift Valley fever virus.

PIPELINESHORT*: Rift valley fever's symptoms are vomiting blood, bleeding from the nose, jaundice, menorrhagia, passing blood in the feces, ecchymoses (caused by bleeding in the skin) and bleeding from venepuncture sites. It often originates from the aedes mosquitoes from Rift

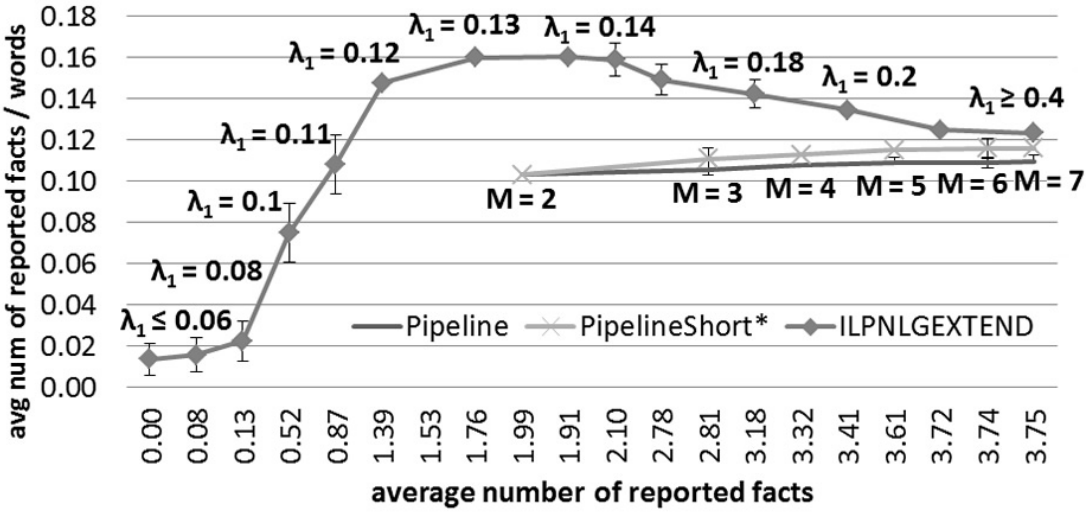


Figure 4.14: Facts per word ratios of ILPNLGEXTEND, PIPELINE, and PIPELINESHORT* for texts generated from the Disease Ontology.

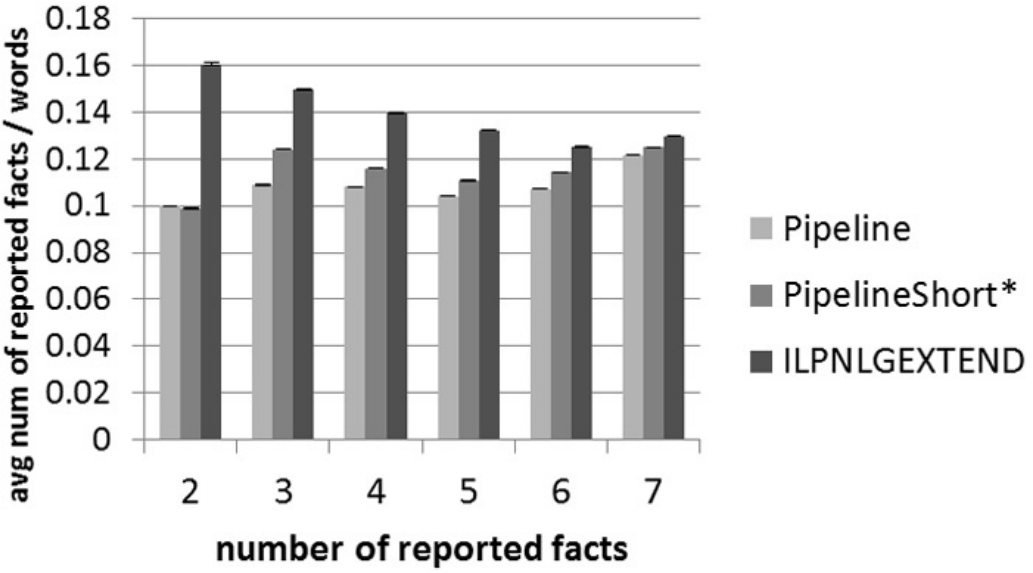


Figure 4.15: Facts per word ratios (grouped by reported facts) of ILPNLGEXTEND, PIPELINE, and PIPELINESHORT* for texts generated from the Disease Ontology.

Valley fever virus.

ILPNLGEXTEND: Rift valley fever is a kind of viral infectious disease. It results in infections. It often originates from the aedes mosquitoes.

4.6.4 Further experiments with the Wine Ontology

The Wine Ontology was further used to test ILPNLGEXTEND with multiple alternative NL names. We added more NL names to the domain-dependent linguistic resources of Section 2.4.1 to ensure that approximately three NL names on average (with a minimum of 2 and a maximum of 5) were available for each one of the individual and classes we generated texts for. We generated texts for the 52 wine individuals and 24 of the wine classes of the Wine Ontology, using PIPELINE, PIPELINESHORT*, and ILPNLGEXTEND.[22] All three systems were allowed to form aggregated sentences with up to $W_{max} = 26$ words.[23] Similarly to Section 4.6.3, W_{max} was the number of words of the longest aggregated sentence in the experiments of Section 2.4.1, where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one. In ILPNLGEXTEND, we used different values for λ_1 ($\lambda_2 = 1 - \lambda_1$), setting $m = 3$, again as in Section 4.6.1. In PIPELINE and PIPELINESHORT*, we used $M = 2,3,4,5,6,7$.[24] For each M, the texts of PIPELINE for the 76 individuals and classes were generated 10 times (not 3, unlike all the previous experiments with PIPELINE); each time, we used one of the different alternative sentence plans for each relation and one of the different alternative NL names for the individual or class the text was being

[22] In the experiments of Section 4.6.1, we had excluded all wine classes, because we could not think of alternative sentence plans for their axioms. The 24 (out of 63) wine classes were included in the experiments of this section, because we were able to provide alternative NL names for them.

[23] Again, we modify PIPELINE and PIPELINESHORT to count words (instead of elements) during aggregation when comparing against ILPNLGEXTEND.

[24] Generating texts for the additional 24 classes required raising the maximum M value to 7, unlike the experiments of Section 4.6.1, where it was 6.

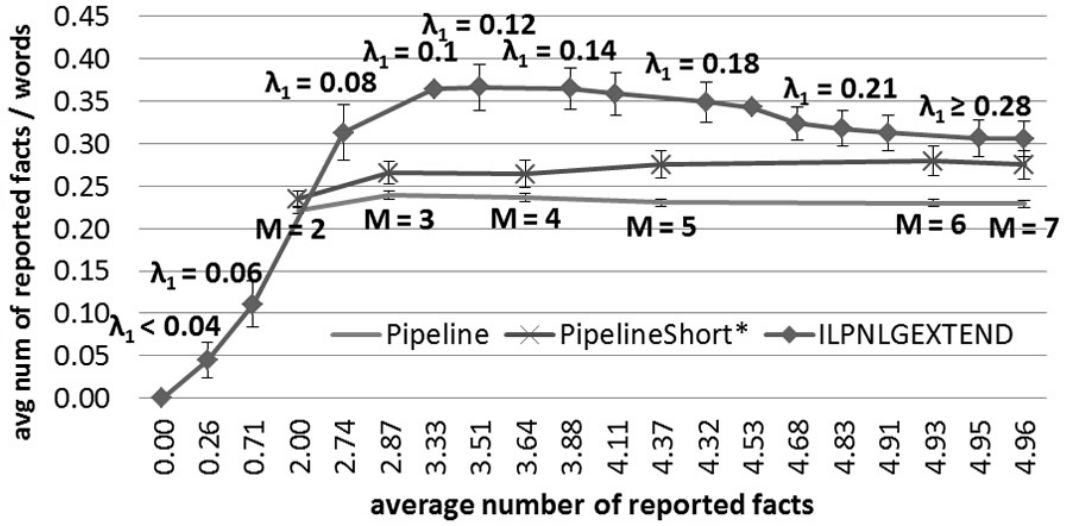


Figure 4.16: Fact per word ratios of the additional experiments with the Wine Ontology.

generated for, since PIPELINE cannot select among alternative NL names (and sentence plans) by itself.

Figures 4.16 and 4.17 show the resulting facts per word ratios, computed in two ways, as in Section 4.6.1.[25] In Fig. 4.16, for $\lambda_1 < 0.04$, ILPNLGEXTEND produces empty texts, because it focuses on minimizing the length of each text. For $\lambda_1 \geq 0.08$, it performs clearly better than the other systems. For $\lambda_1 = 0.12$, it obtains the highest facts per word ratio by selecting the facts and sentence plans that lead to the shortest (in words) aggregated sentences, and NL names that indirectly express facts (thus not requiring separate sentences to express them). For greater values of λ_1, ILPNLGEXTEND selects additional facts whose sentence plans do not aggregate that well or that cannot be indirectly expressed via NL names, which is why the ratio of ILPNLGEXTEND declines. We note that the highest average facts per word ratio of ILPNLGAPPROX (0.37, for $\lambda_1 = 0.12$) of Fig. 4.16 is higher than the highest average ratio (0.33, for $\lambda_1 = 0.3$) we had obtained in Section 4.6.1 with ILPNLG (Fig. 4.3). Also, the overall values of λ_1 are

[25] For $0.28 \leq \lambda_1 \leq 0.9$, ILPNLGEXTEND's results in Fig. 4.16 were identical.

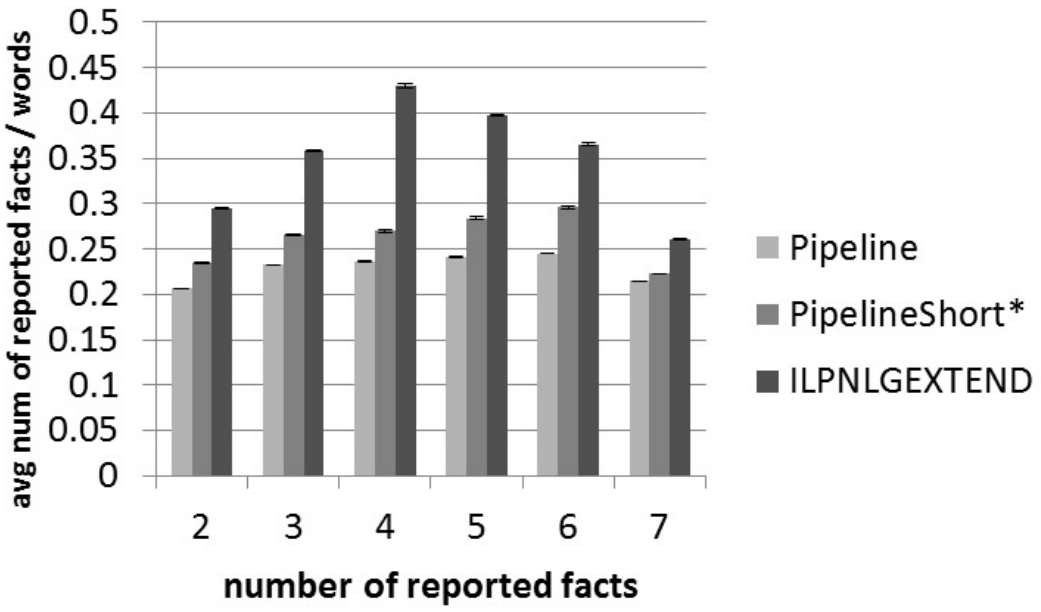


Figure 4.17: Fact per word ratios (grouped by reported facts) of the additional experiments with the Wine Ontology.

now smaller. This is due to the larger number of factors in the right part of the objective function (Eq. 4.13) of ILPNLGEXTEND. Figure 4.17 confirms that ILPNLGEXTEND outperforms the pipelines. In the experiments of this section with ILPNLGEXTEND, the ILP solver was very fast (average: 0.06 sec, worst: 0.64 sec per text).

We show below texts produced by PIPELINE ($M = 4$), PIPELINESHORT* ($M = 4$), and ILPNLGEXTEND ($\lambda_1 = 0.12$). All three texts describe the same wine and report four facts.

PIPELINE: This Sauvignon Blanc is dry and medium. It is made by Stonleigh and it is produced in New Zealand.

PIPELINESHORT*: This delicate tasting and dry Sauvignon Blanc wine originates from New Zealand.

ILPNLGEXTEND: This Stonleigh Sauvignon Blanc is dry, delicate and medium.

ILPNLGEXTEND chose an NL name that avoids expressing the maker of the wine as a separate sentence, and used the same verb ("is") to express the other three facts, which allowed a single aggregated sentence to be formed. It also avoided expressing the origin (New Zealand), which would require a long sentence that would not aggregate well with the others.

4.7 Summary and Contributions of this Chapter

In this chapter, we presented an ILP model that jointly considers decisions in content selection, lexicalization, and sentence aggregation to avoid greedy local decisions and produce more compact texts. An extended version of the ILP model also considers the lengths of alternative NL names and how they can indirectly express facts. We also defined an approximation of our models that generates separately each (possibly aggregated) sentence of the final text and is more efficient when longer texts are generated. The ILP models of this chapter were embedded in NaturalOWL, which used a pipeline architecture in its original form. Experiments with the Wine, Consumer Electronics, and Disease Ontologies confirmed that the ILP models can express more facts per word, with no deterioration in the perceived quality of the generated texts or with improved perceived quality, compared to texts generated by a pipeline architecture. Our experiments also showed that our ILP methods are efficient enough to be used in practice.

The work of this chapter is the first to consider content selection, lexicalization, and sentence aggregation (and a limited form of referring expression generation) as an ILP joint optimization problem in multi-sentence concept-to-text generation. Previous work in NLG either employed a pipeline architecture with greedy local decisions per processing stage, or considered fewer and different processing stages, was concerned with generating single sentences, or had very different inputs.

Chapter 5

The NaturalOWL Implementation

5.1 Introduction

In this chapter we provide an overall presentation of the NaturalOWL system and the NaturalOWL Protégé plug-in, and how they can be used to automatically generate descriptions of individuals and classes of an OWL ontology. We also present some technical details about the implementation of NaturalOWL and its software requirements.

5.2 System design and architecture

The NaturalOWL system has been developed in Java (JDK SE 7) and has been thoroughly tested on Windows XP, Windows Vista, and Windows 7. The NaturalOWL Protégé plug-in has been developed in Java as well, for Protégé version 4.1, but has been tested with later versions as well (up to version 4.3). The plug-in has also been tested on Mac and Linux operating systems. The NaturalOWL version 2 system and all ontologies and respective linguistic resources authored or produced during the work of this thesis are publicly available under a GNU General Public License (GPL).[1]

[1] NaturalOWL version 2 is available from http://nlp.cs.aueb.gr/software.html

Figure 5.1 shows the overall architecture of the NaturalOWL system. The NaturalOWL Protégé plug-in is used to author and edit the ontology that contains the domain-dependent linguistic resources (see Section 2.3) and contains a tab that invokes the generation models of NaturalOWL to generate descriptions. The generation resource production methods (see Sections 3.2.2, 3.2.3 and 3.3.2) are able to produce sentence plans, NL names, lexicon entries (for the adjectives, nouns and verbs that occur in the sentence plans and NL names), and interest scores to avoid redundancy, which are stored in the resources ontology. Both the resource ontology and the domain ontology are inputs to the various generation models of the NaturalOWL system, including the pipeline system (see Section 2.3), and all the variations of the ILP system (see Sections 4.3, 4.4 and 4.5).

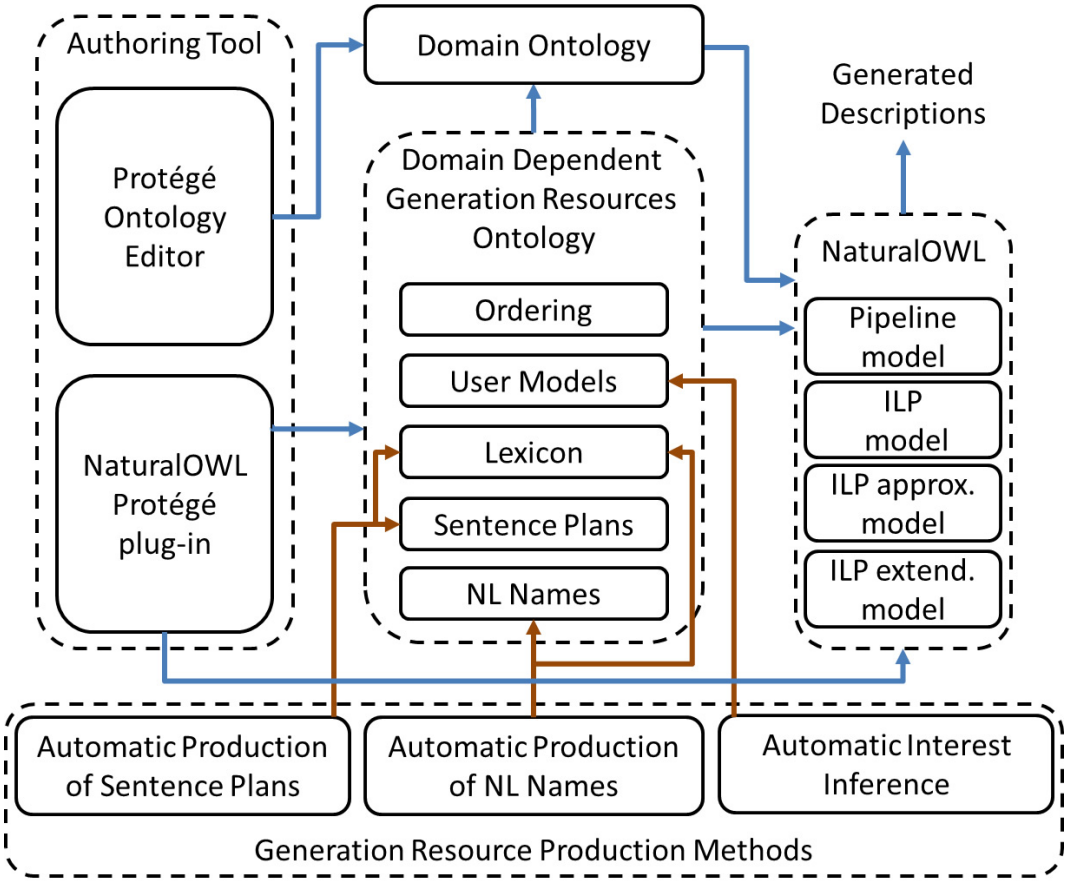


Figure 5.1: Overall architecture of NaturalOWL version 2.

5.3 Use of the NaturalOWL Protégé plug-in

This section presents how an end user can generate descriptions for classes and individuals of an ontology using NaturalOWL. In our examples, we assume the user wishes to generate descriptions from the Wine Ontology, and more specifically for the class `FormanCabernetSauvignon`. All the steps we present here can be applied to any individual or class of any ontology. At this stage, the ontology we examine has not been altered in any way to benefit NaturalOWL and no domain-dependent linguistic resources have been authored for the natural language generation process. We will be using the Protégé ontology editor and our Protégé plug-in that allows authoring the domain-dependent generation resources and invoking NaturalOWL to view the resulting texts; NaturalOWL also provides an application programming interface (API) that allows invoking its methods.

To begin, let us consider the statements concerning `FormanCabernetSauvignon` presented bellow. We present the statements themselves here rather than the corresponding message triples NaturalOWL would convert them to, because this is how the user would view the class through the Protégé editor.

```
EquivalentClasses(:FormanCabernetSauvignon
  ObjectIntersectionOf(:CabernetSauvignon
    ObjectHasValue(:locatedIn :NapaRegion)
    ObjectHasValue(:hasBody :Medium)
    ObjectHasValue(:hasFlavor :Strong)
    ObjectHasValue(:hasSugar :Dry)
    ObjectHasValue(:hasMaker :Forman))
```

To generate a description of `FormanCabernetSauvignon`, we first need to author sentence plans for the properties `locatedIn`, `hasBody`, `hasFlavor`, `hasSugar` and `hasMaker`, and NL names for the individuals `NapaRegion`, `Medium`, `Dry`, `Strong` and `Forman`, and the classes `FormanCabernetSauvignon` and `CabernetSauvignon`.

The reader is reminded that NaturalOWL already provides default sentence plans for domain-independent properties of OWL (e.g., `subclassOf`). The user could skip the steps of authoring linguistic resources and still be able to generate descriptions using the resources NaturalOWL automatically generates by tokenizing the OWL identifiers of the corresponding properties, individuals and classes, but the produced descriptions would be of much inferior text quality (see Section 2.3.2.1).

Figure 5.2 shows how sentence plans can be authored via the "Sentence Plans" tab of NaturalOWL's Protégé plug-in; every box in the figure corresponds to one of the authored sentence plan's slots along with relevant options. A simplified preview of the sentence plan is shown on the top of the tab, while options concerning the authored and all sentence plans overall are available at the bottom. Alternatively, the author may choose to automatically produce sentence plans for a particular property (see Section 3.3.2) via the "New Sentence Plan" button. After the user provides the required input data, sentence plans are produced and presented as in Figure 5.3, by descending confidence of the sentence plan generation method. [2] The user can select which of the produced sentence plans should be added to the linguistic resources. The plug-in also provides example sentences produced by each sentence plan; the user can request more examples if needed.

In a similar way, the user can author, or invoke the automatic method to produce NL names for individuals and classes (see Section 3.2.2). Figures 5.4 and 5.5 show the respective "NL Names" tab of NaturalOWL's Protégé plug-in that is used to author NL names and the results of invoking the automatic NL name generation method. The produced NL names are sorted by decreasing confidence of the NL name generation method. They are accompanied by example sentences where the NL names are used, and copies of the same sentences where the NL name is replaced by a pronoun; this way

[2] We assume that all the documents required by our sentence plan generation method have already been retrieved and preprocessed (e.g., HTML tags have been removed).

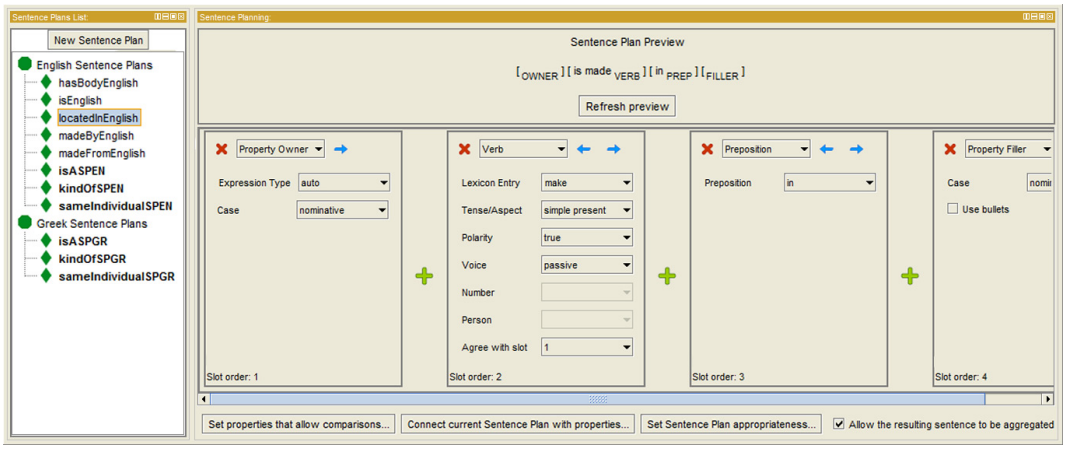


Figure 5.2: Authoring a sentence plan for `locatedIn`.

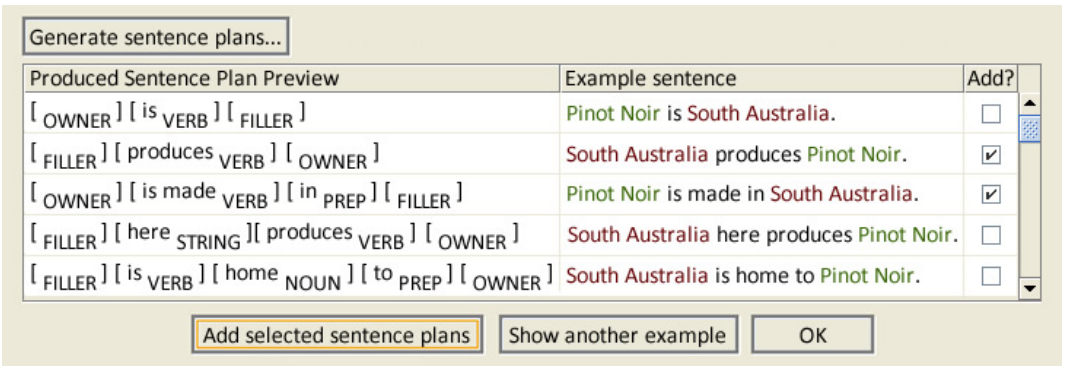


Figure 5.3: Selecting produced sentence plans for `locatedIn`.

the user can check the gender of the produced NL name. If the system infers that the individual or class in question should be anonymized (see Section 3.2.3), it will present this as its first choice amongst the produced NL names.

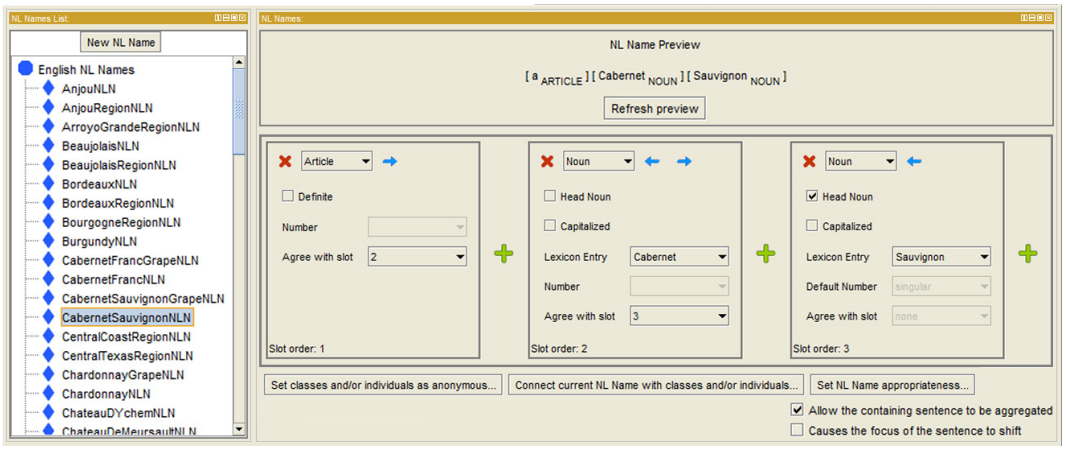


Figure 5.4: Authoring an NL name for `CabernetSauvignon`.

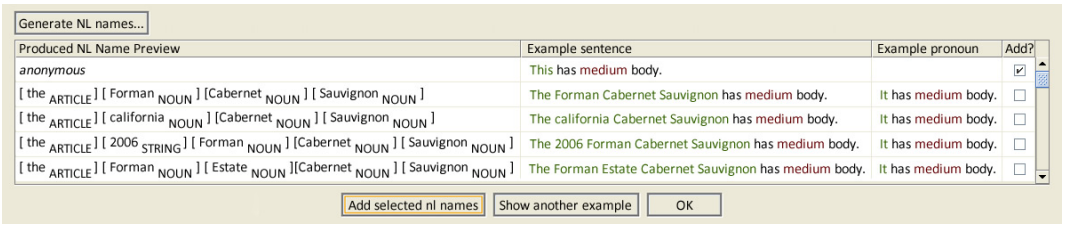


Figure 5.5: Selecting produced NL names for `FormanCabernetSauvignon`.

All the adjectives, nouns, and verbs that are used in the sentence plans and NL names have to be individually authored in the lexicon via the "Lexicon" tab of NaturalOWL's Protégé plug-in. When automatically producing and adding sentence plans and NL names to the linguistic resources, all the necessary lexical entries are also automatically added to the lexicon and will appear in the "Lexicon" tab for review. In Figure 5.6 we show an example for the verb "to make".

Subsequently, the user can create sections, assign properties to them, and edit the order of the sections and the order of the properties in each section (see Section 2.3.1.2),

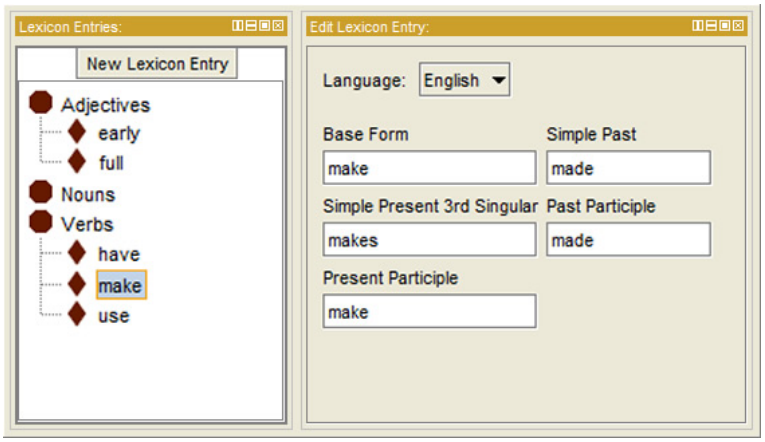


Figure 5.6: Authoring the lexicon entry for the verb "to make".

by using the "Sections and Order" tab of NaturalOWL's Protégé plug-in (Fig. 5.7). The left part of the tab shows the properties that have already been assigned to sections, and the current order of the sections and properties in each section. For example, the property `locatedIn` has been assigned to the `production` section, and has an order of 3. The `production` section has been assigned an order of 20. Each section may also have a label that will act as a title for the corresponding paragraph, if the system is instructed to form paragraphs in the generated descriptions. On the right side of the tab, all the properties that have not yet been assigned to sections (e.g., `hasDrink`) are listed.

In the "User Modelling" tab (Fig. 5.8), the user may edit the preferences of the default "Global User Model" or create and edit custom User Models (see Section 2.3.1.1 and our technical report of the system (Androutsopoulos et al., 2012) for more details). At the bottom of the tab, the user can navigate and select amongst object properties, data properties, domain-independent properties (e.g., `isA`), possible modifiers (e.g., `maxCardinality`) and specific classes or individuals, so that their respective interest and repetition scores can be edited. In our example, for all the message triples involving the property `locatedIn` and the class `DessertWine`, the interest for the adult user type has been set to 1 and the repetitions to 2.

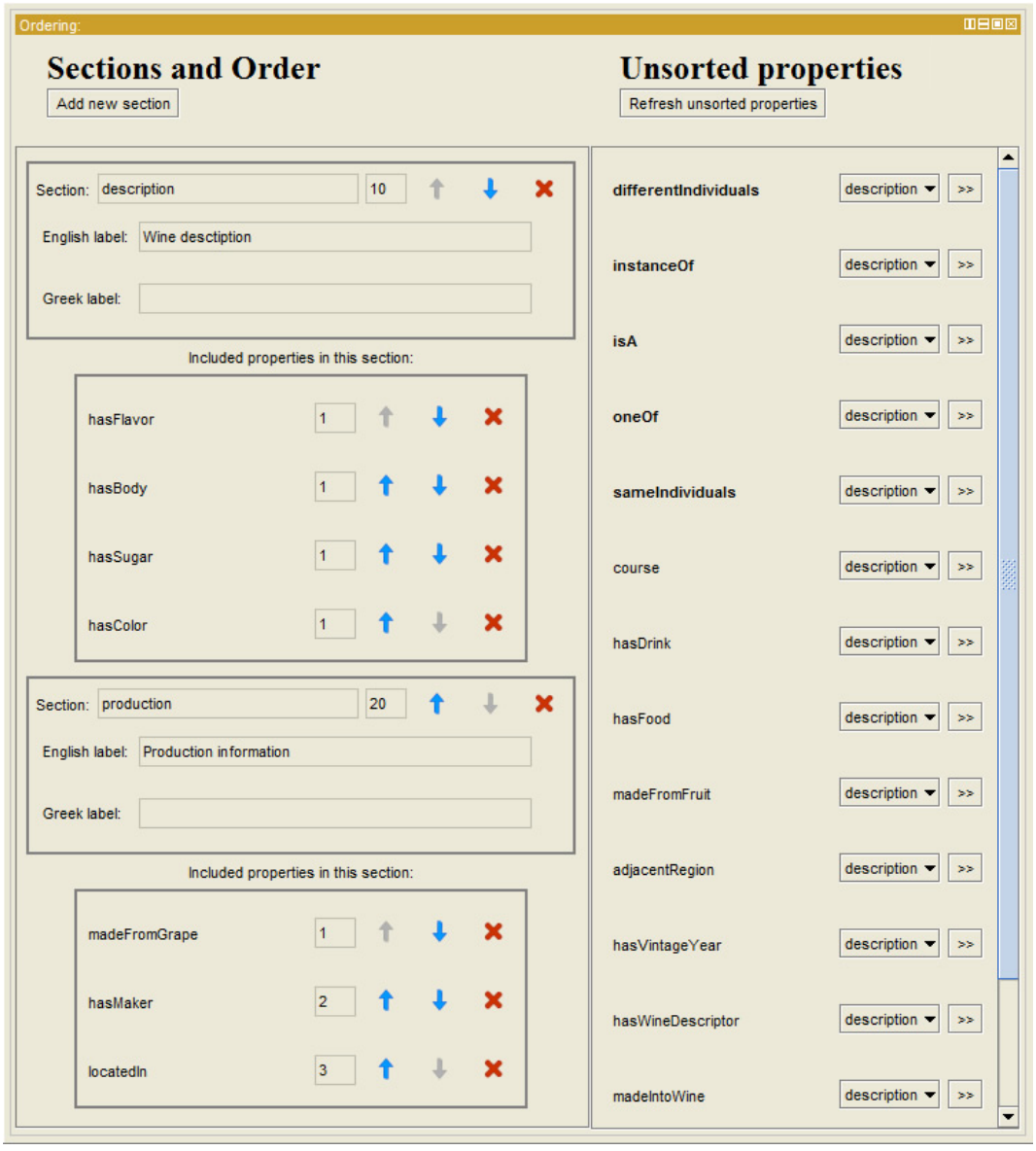


Figure 5.7: Authoring sections and orders.

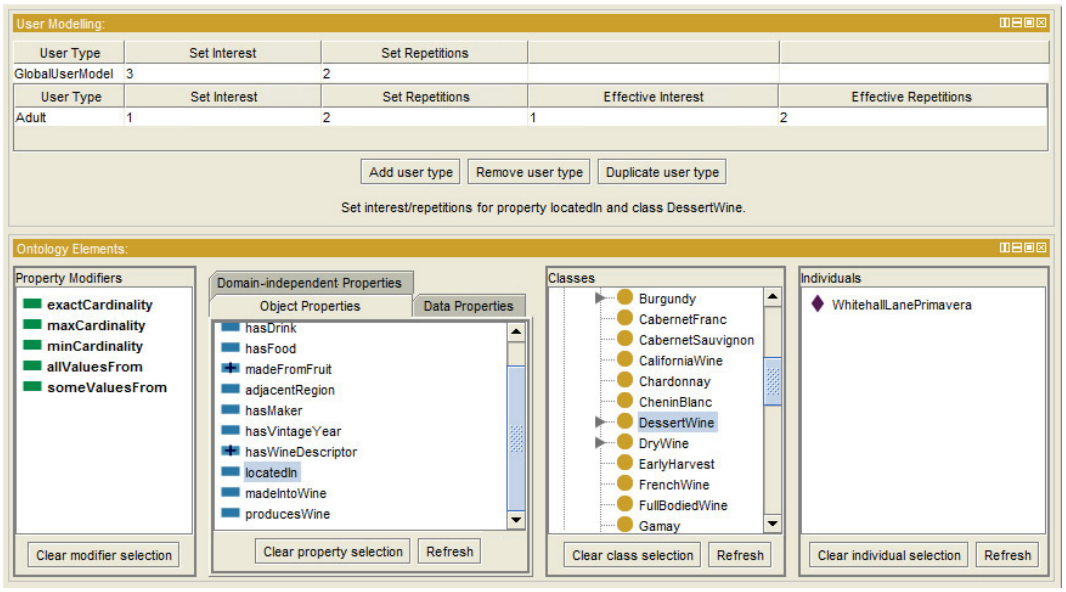


Figure 5.8: Authoring user modelling preferences.

Finally, texts describing classes and individuals can be generated in the "Text Generation" tab of NaturalOWL's Protégé plug-in. Figure 5.9 shows the right side of the tab, where the user can select amongst various options concerning the generation of the text, having first selected a particular class or individual on the left side of the tab (not shown). The desired language (English or Greek) and the target user type are the first choices, followed by the model that will be used to generate the text. There are 4 generation models available to produce the description: the pipeline mode (see Section 2.3), the ILP model (see Section 4.3), the approximation of the ILP model (see Section 4.5), and the extended ILP model (see Section 4.4). Additional options are used to select the graph distance for which the message triples will be retrieved (see Section 2.3.1.1), to impose a limit on each sentence's length (measured in facts, slots, or words), and to manipulate the format of the generated text. For example, the text can be annotated with semantic and syntactic information, as in the following example.

This is a strong, dry Cabernet Sauvignon. It has a medium body. It is made by Forman in the Napa County.

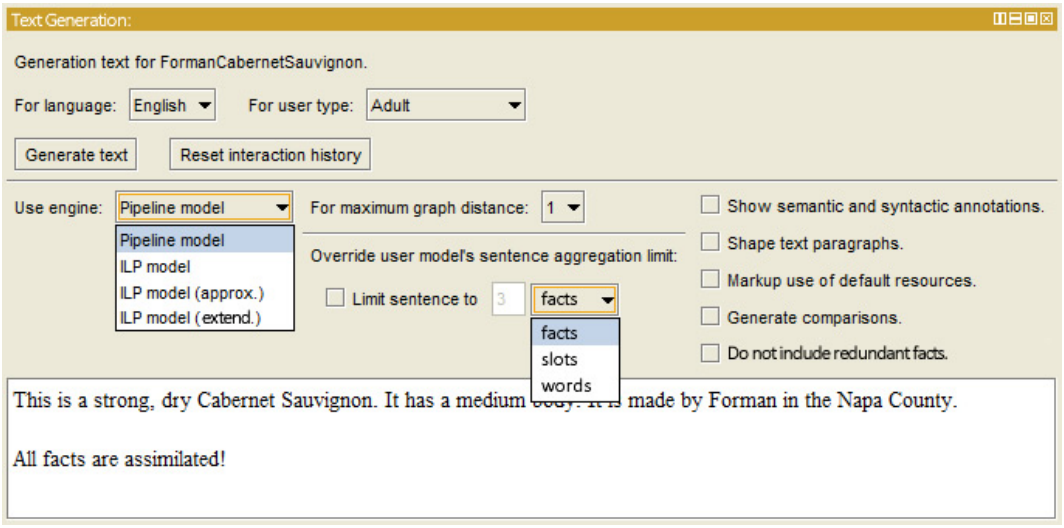


Figure 5.9: Generating a text for `FormanCabernetSauvignon`.

```
<?xml version="1.0" encoding="UTF-8"?>
<AnnotatedText ref="wine:FormanCabernetSauvignon">
    <Period>
        <Sentence Assim="1.0" Interest="1" forProperty="wine:instanceOf">
            <Demonstrative ref="wine:FormanCabernetSauvignon" role="owner">
                This
            </Demonstrative>
            <VERB>is</VERB>
            <NP ref="wine:CabernetSauvignon" role="filler">
                a strong , dry Cabernet Sauvignon
            </NP>
        </Sentence>
        <Punct>.</Punct>
    </Period>
    <Period>
        <Sentence Assim="1.0" Interest="2" forProperty="wine:hasBody">
            <Pronoun ref="wine:FormanCabernetSauvignon" role="owner">It</Pronoun>
            <VERB>has</VERB>
            <TEXT>a</TEXT>
            <NP ref="wine:Medium" role="filler">medium</NP>
        </Sentence>
        <Punct>.</Punct>
    </Period>
    <Period>
```

```
        <Sentence Assim="1.0" Interest="2" forProperty="wine:hasMaker">
            <Pronoun ref="wine:FormanCabernetSauvignon" role="owner">It</Pronoun>
            <VERB>is made</VERB>
            <TEXT>by</TEXT>
            <NP ref="wine:Forman" role="filler">Forman</NP>
            <TEXT>in</TEXT>
            <NP ref="wine:NapaRegion" role="filler">the Napa County</NP>
        </Sentence>
        <Punct>.</Punct>
    </Period>
</AnnotatedText>
```

Alternatively, the generated text can be shaped into paragraphs that correspond to its sections. If titles for the sections have not been provided (as previously shown), their identifiers are used instead.

Wine description

This is a strong, dry Cabernet Sauvignon. It has a medium body.

Production information

It is made by Forman in the Napa County.

The use of default resources (i.e., sentence plans and NL names constructed by tokenizing the corresponding OWL identifiers of properties, classes and individuals) in the generated text can also be marked up. This is primarily used for debugging purposes, in order to locate properties, classes and individuals for which no resources have been authored (or produced). The generated text can also include comparisons (Karakatsiotis, 2007; Melengoglou, 2002) to the previous individuals or classes the user has encountered ("Unlike the previous wine which was red, this wine is white."). The last available option invokes the method that automatically induces interest scores from NL names, to avoid expressing redundant message triples in the text (see Section 3.2.3).

Chapter 6

Conclusions

6.1 Summary of the thesis and its contribution

This thesis centered around NaturalOWL, an open-source NLG system for OWL ontologies, previously developed at AUEB. The system was extensively modified to support OWL 2 and to be able to produce higher quality texts. We also considered text mining and machine learning methods to automatically or semi-automatically extract from the Web NL names and sentence plans, the most important of the domain-dependent linguistic resources that NaturalOWL needs to produce high quality texts. Finally, we aimed to produce more compact texts by developing an Integer Linear Programming model that jointly considers content selection, lexicalization, sentence aggregation, and a limited form of referring expression generation, unlike the pipeline architecture of most natural language generation systems, where the three stages are greedily considered one after the other. Throughout the thesis, ontologies from different domains (e.g., cultural heritage, consumer electronics, bioinformatics) were used.

More precisely, the thesis consists of three main parts. The first part (Chapter 2) provided a detailed description of the new version of NaturalOWL, including its processing stages and its domain-dependent linguistic resources, also comparing the first

version to the original one. By experimenting with three ontologies, we showed that a system like NaturalOWL, which relies on NLG methods to a larger extent than simpler OWL verbalizers, can produce significantly better natural language descriptions of classes and individuals, provided that appropriate domain-dependent generation resources are available. We also showed how the descriptions can be generated in more than one languages (English and Greek), again provided that appropriate resources are available. Subsequent trials measured the extent to which each processing stage and each type of domain-dependent generation resources affects the quality of the generated texts, concluding that NL names, sentence plans, and (to a lesser degree) text plans have the greatest impact on the perceived quality of the generated texts. This part of the thesis constitutes the first detailed discussion of a complete, general-purpose NLG system for OWL ontologies and the particular issues that arise when generating texts from OWL ontologies. The new version of NaturalOWL is the first complete, publicly available NLG system for OWL ontologies, excluding much simpler ontology verbalizers. Furthermore, all the domain-dependent linguistic resources of NaturalOWL are now also represented in OWL, facilitating their publication and sharing on the Web.

Manually authoring domain-specific generation resources is costly and tedious in most NLG systems. In the second part of the thesis (Chapter 3), we proposed methods that employ text mining and machine learning to automatically or semi-automatically extract from the Web sentence plans and NL names, the most important types of domain-specific linguistic resources used by NaturalOWL. Experiments with three ontologies showed that our methods can be used to construct natural language names and sentence plans with minimal human involvement, a few hours at most per ontology. By contrast, manually authoring the same resources is typically a matter of several days. Also, no familiarity with OWL and the inner workings of NaturalOWL are needed for an end-user to produce natural language names and sentence plans using the semi-automatic methods of this part of the thesis. The resulting texts are of almost the same quality

as when employing manually authored generation resources, and much better than texts produced by using generation resources extracted from the relation and entity identifiers of the ontology. Furthermore, unlike previous related work, our methods do not require parallel training corpora of texts and semantic representations, which are very difficult to obtain in practice. Our experiments, however, also indicated that our methods cannot be used on their own in a fully automatic manner, with no human involvement. The methods of this part of the thesis have also been embedded in the new version of NaturalOWL. The processing stages and generation resources of NaturalOWL are typical of NLG systems. Hence, we believe that the work of this part of the thesis is also applicable, at least in principle, to other NLG systems. Our methods may also be useful in simpler ontology verbalizers, where the main concern seems to be to avoid manually authoring domain-specific linguistic resources, currently at the expense of producing texts of much lower quality.

In the third part of the thesis (Chapter 4), we proposed new ILP-based methods that jointly consider the decisions of content selection, lexicalization, and sentence aggregation to avoid the greedy local decisions of a typical pipeline architecture and produce more compact texts, i.e., texts that report more facts per word. Compact texts of this kind are desirable when space is limited or expensive, for example when displaying product descriptions on smartphones, or when including advertisements in Web search results. Experiments with three ontologies confirmed that our ILP methods manage to produce more compact texts, with no deterioration in the perceived quality of the texts or with improved perceived quality, compared to texts generated by a pipeline. We also considered the computational complexity of our methods and their efficiency in practice, and we proposed an approximation of our ILP models that is more efficient when longer texts need to be generated. The work of this part of the thesis is the first to consider content selection, lexicalization, and sentence aggregation (and a limited form of referring expression generation) as an ILP joint optimization problem in multi-sentence

concept-to-text generation. The ILP methods of this part have also been embedded in NaturalOWL. The ontologies, all the software (see also Chapter 5), and the generation resources of the thesis are publicly available.[1]

6.2 Future work

The work of the first part of the thesis could be extended by considering larger scale and more inter-connected ontologies and datasets, especially Linked Data.[2] Most Linked Data currently use only RDF and RDF SCHEMA, but OWL is in effect a superset of RDF SCHEMA and, hence, the work of this thesis is also applicable to Linked Data. A new version of NaturalOWL may be needed, however; for example, the current version stores the domain ontology and all of its instances in memory, which may be impossible with much larger datasets. More elaborate content selection methods may also be necessary with highly inter-connected RDF triples.

The methods of the second part of the thesis, which extract generation resources from the Web, could be improved to be used in a fully automatic manner. For example, dimensionality reduction techniques (e.g., PCA) could be used to improve the classifier of the method that generates sentence plans; features based on dependency trees might also be useful. Further work could aim to produce automatically or semi-automatically text plans or user models. Another future goal would be to produce domain-dependent generation resources for other languages (e.g., Greek); the methods of the second part of this thesis have so far been tested only with English.

The ILP models of the third part of the thesis could be extended to consider additional generation stages (e.g., text planning, or more referring expression generation decisions). It would also be interesting to combine the ILP models with other user modeling components that would assign interest scores to message triples. Another valu-

[1] Consult `http://http://nlp.cs.aueb.gr/software.html`.

[2] See `http://linkeddata.org/`.

able direction would be to examine how to combine the ILP models or, more generally, concept-to-text generation with text snippets extracted from manually written texts.

References

E. Althaus, N. Karamanis, and A. Koller. 2004. Computing locally coherent discourses. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, pages 399–406, Barcelona, Spain.

I. Androutsopoulos and P. Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38(1):135–187.

I. Androutsopoulos, S. Kallonis, and V. Karkaletsis. 2005. Exploiting OWL ontologies in the multilingual generation of object descriptions. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG '05)*, pages 150–155, Aberdeen, UK.

I. Androutsopoulos, J. Oberlander, and V. Karkaletsis. 2007. Source authoring for multilingual generation of personalised object descriptions. *Natural Language Engineering*, 13(3):191–233.

I. Androutsopoulos, G. Lampouras, and D. Galanis. 2012. Generating natural language descriptions from OWL ontologies: A detailed presentation of the NaturalOWL system. Technical report, NLP Group, Department of Informatics, Athens University of Economics and Business, Greece.

I. Androutsopoulos, G. Lampouras, and D. Galanis. 2013. Generating natural language descriptions from OWL ontologies: the NaturalOWL system. *Journal of Artificial Intelligence Research*, 48(1):671–715.

G. Angeli, P. Liang, and D. Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*, pages 502–512, Cambridge, MA.

G. Antoniou and F. van Harmelen. 2008. *A Semantic Web primer*. MIT Press, 2nd edition.

C. Areces, A. Koller, and K. Striegnitz. 2008. Referring expressions as formulas of description logic. In *Proceedings of the 5th International Natural Language Generation Conference (INLG '08)*, pages 42–49, Salt Fork, OH.

F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. 2002. *The Description Logic Handbook*. Cambridge University Press.

R. Barzilay and M. Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP '05)*, pages 331–338, Vancouver, British Columbia, Canada.

R. Barzilay and M. Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Proceedings of the Conference on Human Language Technology and the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT '06)*, pages 359–366, New York, NY.

R. Barzilay and M Lapata. 2008. Modeling local coherence: an entity-based approach. *Computational Linguistics*, 34(1):1–34.

R. Barzilay and L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation & summarization. In *Proceedings of the 43rd Annual Meeting of the Association of Computational Linguistics (ACL '04)*, pages 113–120, Ann Arbor, MI.

R. Barzilay, N. Elhadad, and K. McKeown. 2002. Inferring strategies for sentence ordering in multidocument. news summarization. *Journal of Artificial Intelligence Research*, 17(1):35–55.

J.A. Bateman. 1990. Upper modelling: A general organisation of knowledge for natural language processing. In *Proceedings of the 5th International Workshop on Natural Language Generation (INLGW '90)*, pages 54–61, Dawson, PA.

J.A. Bateman. 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3(1):15–56.

A. Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

T. Berg-Kirkpatrick, D. Gillick, and D. Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT'11)*, pages 481–490, Portland, Oregon.

R. Bernardi, D. Calvanese, and C. Thorne. 2007. Lite natural language. In *7th Intenational Workshop on Computational Semantics (IWCS-7)*, Tilburg, The Netherlands.

T. Berners-Lee, J. Hendler, and O. Lassila. 2001. The Semantic Web. *Scientific American*, May(1):34–43.

K. Bontcheva and H. Cunningham. 2003. The Semantic Web: a new opportunity and challenge for human language technology. In *Proceedings of the Workshop on Human Language Technologies for the Semantic Web and Web Services, 2nd International Semantic Web Conference (ISWC '03)*, Sanibel Island, FL.

K. Bontcheva and Y. Wilks. 2004. Automatic report generation from ontologies: the MIAKT approach. In *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB '04)*, pages 324–335, Manchester, UK.

K. Bontcheva, V. Tablan, D. Maynard, and H. Cun-ningham. 2004. Evolving GATE to meet new challenges in language engineering. *Natural Language Engineering*, 10(3–4):349–373.

K. Bontcheva. 2005. Generating tailored textual summaries from ontologies. In *Proceedings of the 2nd European Semantic Web Conference (ESWC '05)*, pages 531–545, Heraklion, Greece.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117.

S. Busemann and H. Horacek. 1998. A flexible shallow approach to text generation. In *Proceedings of the 9th International Workshop on Natural Language Generation (INLG '98)*, pages 238–247, New Brunswick, NJ.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *In Proceedings of the 25th International Conference on Machine Learning (ICML '08)*, pages 128–135, Helsinki, Finland.

H. Chen, S.R.K. Branavan, R. Barzilay, and D.R. Karger. 2009. Content modeling using latent permutations. *Journal of Artificial Intelligence Research*, 36(1):129–163.

H. Cheng and C. Mellish. 2000. Capturing the interaction between aggregation and text planning in two generation systems. In *1st International Conference on Natural Language Generation (INLG '00)*, pages 186–193, Mitzpe Ramon, Israel.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

J. Clarke and M. Lapata. 2008. Global inference for sentence compression: an integer linear programming approach. *Journal of Artificial Intelligence Research*, 1(31):399–429.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '02)*, pages 1–8, Stroudsburg, PA, USA.

A. Cregan, R. Schwitter, and T. Meyer. 2007. Sydney OWL syntax – towards a controlled natural language syntax for OWL. In *Proceedings of the OWL: Experiences and Directions Workshop (OWLED '07)*, Innsbruck, Austria.

R. Dale, S.J. Green, M. Milosavljevic, C. Paris, C. Verspoor, and S. Williams. 1998. Dynamic document delivery: generating natural language texts on demand. In *Proceedings of the 9th International Conference and Workshop on Database and Expert Systems Applications (DEXA '98)*, pages 131–136, Vienna, Austria.

H. Dalianis. 1999. Aggregation in natural language generation. *Computational Intelligence*, 15(4):384–414.

L. Danlos. 1984. Conceptual and linguistic decisions in generation. In *Proceedings of the 10th International Conference on Computational Linguistics (COLING '84)*, pages 501–504, Stanford, CA.

D. Dannells. 2012. On generating coherent multilingual descriptions of museum objects from Semantic Web ontologies. In *Proceedings of the 7th International Natural Language Generation Conference (INLG'12)*, pages 76–84, Utica, IL.

D. Dannels. 2008. Generating tailored texts for museum exhibits. In *Proceedings of the 6th edition of the Language Resources and Evaluation Conference, Workshop on Language Technology for Cultural Heritage Data (LREC/LaTeCH '08)*, pages 17–20, Marrakech, Morocco.

G. B. Dantzig. 1963. *Linear Programming and Extensions*. Princeton University Press.

B. Davis, A.A. Iqbal, A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, and S. Handschuh. 2008. Roundtrip ontology authoring. In *Proceedings of the 7th International Conference on the Semantic Web (I-Semantics '11)*, pages 50–65, Karlsruhe, Germany.

S. Demir, S. Carberry, and K.F. McCoy. 2010. A discourse-aware graph-based content-selection framework. In *Proceedings of the 6th International Natural Language Generation Conference (INLG'10)*, pages 17–25, Trim, Co. Meath, Ireland.

R. Denaux, V. Dimitrova, A. Cohn, C. Dolbear, and G. Hart. 2010. Rabbit to OWL: Ontology authoring with a CNL-based tool. In *Controlled Natural Language*, volume 5972, pages 246–264. Springer.

R. Denaux, C. Dolbear, G. Hart, V. Dimitrova, and A. G. Cohn. 2011. Supporting domain experts to construct conceptual ontologies. *Web Semantics*, 9(2):113–127.

P.A Duboue and K.R. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of the 39th Meeting of the Annual Meeting of the Association for Computational Linguistics (ACL '01)*, pages 172–179, Toulouse, France.

P.A. Duboue and K.R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '03)*, pages 121–128, Sapporo, Japan.

D. Duma and E. Klein. 2013. Generating natural language from Linked Data: Unsupervised template extraction. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS'13)*, pages 83–94, Potsdam, Germany.

M. Elhadad and J. Robin. 1996. SURGE: A reusable comprehensive syntactic realization component. In *Proceedings of the 8th International Natural Language Generation Workshop (Posters and Demonstrations) (INLG '96)*, pages 1–4, Herstmonceux Castle, Sussex, UK.

M. Elsner, J. Austerweil, and E. Charniak. 2007. A unified local and global model for discourse coherence. In *Proceedings of the Human Language Technologies Conference of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT '07)*, pages 436–443, Rochester, New York.

M. Evaggelakaki. 2014. Generation of natural language texts from biomedical ontologies with the NaturalOWL system. BSc thesis, Department of Informatics, Athens University of Economics and Business, in Greek.

C. Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, B. Davis, and S. Handschuh. 2007. CLOnE: Controlled language for ontology editing. In *Proceedings of the 6th International Semantic Web and 2nd Asian Semantic Web Conference (ASWC/ISWC '07)*, pages 142–155, Busan, Korea.

D. Galanis and I. Androutsopoulos. 2007. Generating multilingual descriptions from linguistically annotated OWL ontologies: the NaturalOWL system. In *Proceedings of the 11th European Workshop on Natural Language Generation (ENLG '07)*, pages 143–146, Schloss Dagstuhl, Germany.

D. Galanis, G. Karakatsiotis, G. Lampouras, and I. Androutsopoulos. 2009. An open-source natural language generator for OWL ontologies and its use in Protégé and Second Life. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (demos) (EACL '09)*, pages 17–20, Athens, Greece.

D. Galanis, G. Lampouras, and I. Androutsopoulos. 2012. Extractive multi-document summarization with ILP and Support Vector Regression. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING'12)*, pages 911–926, Mumbai, India.

A. Gatt and E. Reiter. 2009. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG '09)*, pages 90–93, Athens, Greece.

B.C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. 2008. OWL 2: The next step for OWL. *Web Semantics*, 6(4):309–322.

B.J. Grosz, A.K. Joshi, and S. Weinstein. 1995. Centering: a framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

S. Gupta and C. D. Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the 18th Conference on Computational Natural Language Learning (CoNLL'14)*, pages 98–108, Ann Arbor, Michigan.

V. Haarslev and R. Moller. 2001. RACER system description. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR '01)*, pages 701–706, Siena, Italy.

C. Halaschek-Wiener, J. Golbeck, B. Parsia, V. Kolovski, and J. Hendler. 2008. Image browsing and natural language paraphrases of semantic web annotations. In *Proceedings of the 1st International Workshop on Semantic Web Annotations for Multimedia (SWAMM '06)*, Tenerife, Spain.

C. Hallett, D. Scott, and R. Power. 2007. Composing questions through conceptual authoring. *Computational Linguistics*, 33(1):105–133.

M. Halliday. 1994. *Introduction to Functional Grammar*. Edward Arnold, 2nd edition.

I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. 2003. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics*, 1(1):7–26.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL '07)*, pages 144–151, Prague, Czech Republic.

A. Isard, J. Oberlander, I. Androutsopoulos, and C. Matheson. 2003. Speaking the users' languages. *IEEE Intelligent Systems*, 18(1):40–45.

D. Jurafsky and J. Martin. 2008. *Speech and Language Processing*. Prentice Hall.

K. Kaljurand and N. Fuchs. 2007. Verbalizing OWL in Attempto Controlled English. In *Proceedings of the 3rd International Workshop on OWL: Experiences and Directions (OWLED '07)*, Innsbruck, Austria.

K. Kaljurand. 2007. *Attempto Controlled English as a Semantic Web Language*. Ph.D. thesis, Faculty of Mathematics and Computer Science, University of Tartu, Estonia.

G. Karakatsiotis. 2007. Automatic generation of comparisons in a natural language generation system. MSc thesis, Department of Informatics, Athens University of Economics and Business, in Greek.

N. Karamanis, M. Pesio, C. Mellish, and J. Oberlander. 2009. Evaluating Centering for Information Ordering using Corpora. *Computational Linguistics*, 35(1):29–46.

RichardM. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US.

R. Kasper and R. Whitney. 1989. SPL: A sentence plan language for text generation. Technical report, Information Sciences Institute, University of Southern California.

E. Kaufmann and A. Bernstein. 2010. Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. *Web Semantics*, 8(4):377–393.

C. Kelly, A. Copestake, and N. Karamanis. 2009. Investigating content selection for language generation using machine learning. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG '09)*, pages 130–137, Athens, Greece.

Dan Klein and Chris Manning. 2002. Parsing and hypergraphs. In Harry Bunt, John Carroll, and Giorgio Satta, editors, *New Developments in Parsing Technology*, pages 351–372. Kluwer Academic Publishers.

I. Konstas and M. Lapata. 2012a. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association of Computational Linguistics (ACL'12)*, pages 369–378, Jeju Island, Korea.

I. Konstas and M. Lapata. 2012b. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the Conference on Human Language Technology of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT'12)*, pages 752–761, Montréal, Canada.

E. Krahmer and K. van Deemter. 2012. Computational generation of referring expressions: A survey. *Comput. Linguistics*, 38(1):173–218.

P. Kuznetsova, V. Ordonez, A. Berg, T. Berg, and Y. Choi. 2012. Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association of Computational Linguistics (ACL'12)*, pages 359–368, Jeju Island, Korea.

G. Lampouras and I. Androutsopoulos. 2013a. Using integer linear programming for content selection, lexicalization, and aggregation to produce compact texts from owl ontologies. In *Proceedings of the 14th European Workshop on Natural Language Generation (ENLG'13)*, pages 51–60, Sofia, Bulgaria.

G. Lampouras and I. Androutsopoulos. 2013b. Using integer linear programming in concept-to-text generation to produce more compact texts. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (ACL'13)*, pages 561–566, Sofia, Bulgaria.

I. Langkilde. 2000. Forest based statistical sentence generation. In *Proceedings of the 1st Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '00)*, pages 170–177, Seattle, WA.

B. Lavoie and O. Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP '97)*, pages 265–268, Washington DC.

Z. Li and J. Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing of the Association of Computational Linguistics (ACL/EMNLP '09)*, pages 40–51, Singapore.

P. Liang, M. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the 47th Meeting of the Association of Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/AFNLP '09)*, pages 91–99, Suntec, Singapore.

S.F. Liang, D. Scott, R. Stevens, and A. Rector. 2011a. Unlocking medical ontologies for non-ontology experts. In *Proceedings of the 10th Workshop on Biomedical Natural Language Processing (BioNLP '11)*, pages 174–181, Portland, OR.

S.F. Liang, R. Stevens, D. Scott, and A. Rector. 2011b. Automatic verbalisation of SNOMED classes using OntoVerbal. In *Proceedings of the 13th Conference on Artificial Intelligence in Medicine (AIME'11)*, pages 338–342, Bled, Slovenia.

W. Lu and H. T. Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*, pages 1611–1622, Edinburgh, United Kingdom.

W. Lu, H. T. Ng, W. S. Lee, and L. S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 783–792, Honolulu, Hawaii.

W. Lu, H. T. Ng, and W. S. Lee. 2009. Natural language generation with tree conditional random fields. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '09)*, pages 400–409, Singapore.

W.C. Mann and S.A. Thompson. 1998. Rhetorical structure theory: A theory of text organization. *Text*, 8(3):243–281.

C.D. Manning and H. Schutze. 2000. *Foundations of Statistical Natural Language Processing*. MIT Press.

T. Marciniak and M. Strube. 2005. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL '05)*, pages 136–143, Ann Arbor, MI.

R. McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the European Conference on Information Retrieval (ECIR '07)*, pages 557–564, Rome, Italy.

K.R. McKeown. 1985. *Text Generation*. Cambridge Univ. Press.

S.W. McRoy, S. Channarukul, and S.S. Ali. 2003. An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381–420.

A. Melengoglou. 2002. Multilingual aggregation in the M-PIRO system. Master's thesis, School of Informatics, University of Edinburgh, UK.

C. Mellish and J.Z. Pan. 2008. Natural language directed inference from ontologies. *Artificial Intelligence*, 172(10):1285–1315.

C. Mellish and X. Sun. 2006. The Semantic Web as a linguistic resource: opportunities for natural language generation. *Knowledge Based Systems*, 19(5):298–303.

C. Mellish, D. Scott, L. Cahill, D. Paiva, R. Evans, and M. Reape. 2006a. A reference architecture for natural language generation systems. *Natural Language Engineering*, 12(1):1–34.

C. Mellish, D. Scott, L. Cahill, D. Paiva, R. Evans, and M. Reape. 2006b. A reference architecture for natural language generation systems. *Natural Language Engineering*, 12(1):1–34.

C. Mellish. 2010. Using Semantic Web technology to support NLG – case study: OWL finds RAGS. In *Proceedings of the 6th International Natural Language Generation Conference (INLG'10)*, pages 85–93, Trim, Co. Meath, Ireland.

B. Motik, R. Shearer, and I. Horrocks. 2007. Optimized reasoning in description logics using hypertableaux. In *Proceedings of the 21st International Conference on Automated Deduction (CADE '07)*, pages 67–83, Bremen, Germany.

I. Muslea. 1999. Extraction patterns for information extraction tasks. In *Proceedings of the AAAI Workshop on Machine Learning for Information Extraction*, pages 1–6, Orlando, Florida.

A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann, and D. Gerber. 2013. Sorry, I don't speak SPARQL: Translating SPARQL queries into natural language. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'13)*, pages 977–988, Rio de Janeiro, Brazil.

T.A.T. Nguyen, R. Power, P. Piwek, and S. Williams. 2012. Planning accessible explanations for entailments in OWL ontologies. In *Proceedings of the 7th International Natural Language Generation Conference (INLG'12)*, pages 110–114, Utica, IL.

J. Oberlander, G. Karakatsiotis, A. Isard, and I. Androutsopoulos. 2008. Building an adaptive museum gallery in Second Life. In *Proceedings of Museums and the Web*, Montreal, Canada.

M. O'Donnell, C. Mellish, J. Oberlander, and A. Knott. 2001. ILEX: an architecture for a dynamic hypertext generation system. *Natural Language Engineering*, 7(3):225–250.

S. Patwardhan and E. Riloff. 2006. Learning domain-specific information extraction patterns from the web. In *Proceedings of the Association for Computational Linguistics Workshop on Information Extraction Beyond the Document*, pages 66–73, Sydney, Australia.

M. Poesio, R. Stevenson, and B. Di Eugenio. 2004. Centering: A parameter theory and its instantiations. *Computational Linguistics*, 30(3):309–363.

R. Power and D. Scott. 1998. Multilingual authoring using feedback texts. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Meeting of the Association of Computational Linguistics (COLING/ACL '98)*, pages 1053–1059, Montreal, Canada.

R. Power and A. Third. 2010. Expressing OWL axioms by English sentences: Dubious in theory, feasible in practice. In *Proceedings of the 23rd International Conference on Computional Linguistics (COLING'10)*, pages 1006–1013, Beijing, China.

R. Power. 2009. Towards a generation-based semantic web authoring tool. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG '09)*, pages 9–15, Athens, Greece.

R. Power. 2010. Complexity assumptions in ontology verbalisation. In *Proceedings of the 48th Annual Meeting of the Association of Computational Linguistics (short papers) (ACL'10)*, pages 132–136, Uppsala, Sweden.

R. Power. 2011. Deriving rhetorical relationships from semantic content. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG '11)*, pages 82–90, Nancy, France.

R. Power. 2012. OWL simplified english: a finite-state language for ontology editing. In *Proceedings of the 3rd International Workshop on Controlled Natural Language (CNL'12)*, pages 44–60, Zurich, Switzerland.

A Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of the 1st Annual Conference of the North American Chapter of the Association for Computational Linguistics (ACL '00)*, pages 194–201, Seattle, WA.

A Ratnaparkhi. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech & Language*, 16(3-4):435–455.

A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. 2004. OWL pizzas: Practical experience of teaching OWL-DL: Common errors and common patterns. In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW '04)*, pages 63–81, Northamptonshire, UK.

E. Reiter and R. Dale. 2000. *Building Natural Language Generation systems*. Cambridge University Press.

E. Reiter. 1995. NLG vs. templates. In *Proceedings of the 5th European Workshop on Natural Language Generation*, Leiden, The Netherlands.

Y. Ren, K. van Deemter, and J.Z. Pan. 2010. Charting the potential of description logic for the generation of referring expressions. In *Proceedings of the 6th International Natural Language Generation Conference (INLG'10)*, pages 115–123, Trim, Co. Meath, Ireland.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence (AAAI/IAAI '99)*, pages 474–479, Orlando, Florida, USA.

E. Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence - Volume 2 (AAAI '96)*, pages 1044–1049, Portland, Oregon.

Alexander Schrijver. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc.

N. Schutte. 2009. Generating natural language descriptions of ontology concepts. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG '09)*, pages 106–109, Athens, Greece.

R. Schwitter and M. Tilbrook. 2004. Controlled natural language meets the Semantic Web. In *Proceedings of the Australasian Language Technology Workshop (ALTW '04)*, pages 55–62, Sydney, Australia.

R. Schwitter, K. Kaljurand, A. Cregan, C. Dolbear, and G. Hart. 2008. A comparison of three controlled natural languages for OWL 1.1. In *Proceedings of the 4th OWL: Experiences and Directions Workshop (OWLED '08)*, Washington DC.

R. Schwitter. 2010a. Controlled natural languages for knowledge representation. In *Proceedings of the 23rd International Conference on Computational Linguistics (posters) (COLING'10)*, pages 1113–1121, Beijing, China.

R. Schwitter. 2010b. Creating and querying formal ontologies via controlled natural language. *Applied Artificial Intelligence*, 24(1-2):149–174.

N. Shadbolt, T. Berners-Lee, and W. Hall. 2006. The Semantic Web revisited. *IEEE Intell. Systems*, 21(3):96–101.

E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5(2):51–53.

R. Stevens, J. Malone, S. Williams, R. Power, and A. Third. 2011. Automatic generation of textual class definitions from OWL to English. *Biomedical Semantics*, 2(S 2:S5).

H. Szpektor, I.and Tanev, I. Dagan, and B. Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '04)*, pages 41–48, Barcelona, Spain.

A. Third. 2012. "Hidden semantics": What can we learn from the names in an ontology? In *Proceedings of the 7th International Natural Language Generation Conference (INLG'12)*, pages 67–75, Utica, IL.

S. Thomaidou, I. Lourentzou, P. Katsivelis-Perakis, and M. Vazirgiannis. 2013. Automated snippet generation for online advertising. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management (CIKM'13)*, pages 1841–1844, San Francisco, California, USA.

S. Thomaidou. 2014. *Automated Creation and Optimization of Online Advertising Campaigns*. Ph.D. thesis, Department of Informatics, Athens University of Economics and Business.

D. Tsarkov and I. Horrocks. 2006. FaCT++ description logic reasoner: System description. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR '06)*, pages 292–297, Seattle, WA.

K. van Deemter, E. Krahmer, and M. Theune. 2005. Real versus template-based natural language generation: a false opposition? *Computational Linguistics*, 31(1):15–24.

M.A. Walker, O. Rambow, and M. Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the 2nd Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '01)*, pages 17–24, Pittsburgh, PA.

S. Wan, M. Dras, R. Dale, and C. Paris. 2010. Spanning tree approaches for statistical sentence generation. In *Empirical Methods in Natural Language Generation*, pages 13–44. Springer.

M. White. 2006. CCG chart realization from disjunctive inputs. In *Proceedings of the 4th International Natural Language Generation Conference (INLG '06)*, pages 12–19, Sydney, Australia.

S. Williams and R. Power. 2010. Grouping axioms for more coherent ontology descriptions. In *Proceedings of the 6th International Natural Language Generation Conference (INLG '10)*, pages 197–201, Trim, Co. Meath, Ireland.

S. Williams, A. Third, and R. Power. 2011. Levels of organization in ontology verbalization. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG '11)*, pages 158–163, Nancy, France.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Conference on Human Language Technology of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL '06)*, pages 439–446, New York, New York.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pages 960–967, Prague, Czech Republic.

K. Woodsend and M. Lapata. 2012. Multiple aspect summarization using ILP. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '12)*, pages 233–243, Jesu Island, Korea.

F. Xu, H. Uszkoreit, and H. Li. 2007. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pages 584–591, Prague, Czech Republic.